e-Informatica

software engineering journal

2019 volume 13 issue 1

e-Informatica

e-Informatica

software engineering journal

2019 volume 13 issue 1

e-Informatica

Wrocław University
of Science and Technology

Editors
Zbigniew Huzar (*Zbigniew.Huzar@pwr.edu.pl*)
Lech Madeyski (*Lech.Madeyski@pwr.edu.pl, http://madeyski.e-informatyka.pl*)

Department of Software Engineering, Faculty of Computer Science and Management,
Wrocław University of Science and Technology, 50-370 Wrocław, Wybrzeże Wyspiańskiego 27, Poland

e-Informatica Software Engineering Journal
www.e-informatyka.pl, DOI: 10.5277/e-informatica

Editorial Office Manager: Wojciech Thomas

Typeset by Wojciech Myszka with the LaTeX 2_ε Documentation Preparation System

OFICYNA WYDAWNICZA POLITECHNIKI WROCŁAWSKIEJ
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław
www.oficyna.pwr.edu.pl;
e-mail: oficwyd@pwr.edu.pl; zamawianie.ksiazek@pwr.edu.pl

ISSN 1897-7979

Print and binding: beta-druk, www.betadruk.pl

Editorial Board

Contents

e-Informatica Software Engineering Journal, Volume 13, Issue 1, 2019, pages: 7–35, DOI 10.5277/e-Inf190101

Usage, Retention, and Abandonment of Agile Practices: A Survey and Interviews Results

Indira Nurdiani*, Jürgen Börstler**, Samuel Fricker***, Kai Petersen****

*Department of Software Engineering/DTU Compute – Software and Process Engineering Section, Blekinge Institute of Technology, Sweden/Technical University of Denmark, Denmark

**Department of Software Engineering, Blekinge Institute of Technology, Sweden

***Institute for Interactive Technologies, Fachhochschule Nordwestschweiz, Switzerland

****Chair for Software Engineering, University of Applied Sciences Flensburg, Germany

indira.nurdiani@bth.se/innu@dtu.dk, jurgen.borstler@bth.se, samuel.fricker@fhnw.ch, kai.petersen@hs-flensburg.de

Abstract

Background: A number of Agile maturity models (AMMs) have been proposed to guide software organizations in their adoption of Agile practices. Typically the AMMs suggest that higher maturity levels are reached by gradually adding more practices. However, recent research indicates that certain Agile practices, like test-driven development and continuous integration, are being abandoned. Little is known on the rationales for abandoning Agile practices.
Aim: We aim to identify which Agile practices are abandoned in industry, as well as the reasons for abandoning them.
Method: We conducted a web survey with 51 respondents and interviews with 11 industry practitioners with experience in Agile adoption to investigate why Agile practices are abandoned.
Results: Of the 17 Agile practices that were included in the survey, all have been abandoned at some point. Nevertheless, respondents who retained all practices as well as those who abandoned one or more practices, perceived their overall adoption of Agile practices as successful.
Conclusion: Going against the suggestions of the AMMs, i.e. abandoning Agile one or more practices, could still lead to successful outcomes. This finding indicates that introducing Agile practices gradually in a certain order, as the AMMs suggest, may not always be suitable in different contexts.

Keywords: Agile practices, Agile maturity models, survey

1. Introduction

The software industry is highly competitive. Agile methods, like Scrum and eXtreme Programming (XP), help to tackle the challenges of rapid changes in the environment of software organizations and help to reduce time to market, minimize development costs, and improve software quality [1]. Agile practices are the enactment of Agile principles [2].

A recent survey indicates that some practices like *test-driven development (TDD)*, *pair programming*, and *continuous integration* are being abandoned [3]. Abandoning Agile practices seems contradictory to common guidelines such as Agile maturity models (AMMs) [4–6] that prescribe which practices should be implemented and when according to certain maturity levels. According to the AMMs, the more mature an organization becomes, the more Agile practices are adopted. However, the indication of abandonment of practices could also be due to lack of guidance. Perhaps such practices were not introduced at the right time, given the maturity of the software development teams or organization, because Agile practices dependencies are not well known.

Submitted: 6 February 2018; Revised: 17 July 2018; Accepted: 17 July 2018; Available online: 30 September 2018

Table 1. Allocation of Agile practices to maturity levels in three AMMs

	Sidky et al. [6]	Patel & Ramachandran [5]	Nawrocki et al. [4]
Context	Agile practice adoption based on a measurement index	Agile practice adoption based on CMM(I)	Adoption of XP based on other maturity models
Level 1	On-site customer, collaborative planning, coding standard	–	–
Level 2	Tracking progress, continuous delivery	Tracking progress, on-site customer, planning game, TDD	Planning game, collaborating customer (on-site customer), user stories, metaphors
Level 3	F2F meeting, continuous integration, self-organizing team	Refactoring, pair programming, continuous integration, TDD, coding standard, collective ownership	Pair programming, coding standard, collective ownership, continuous integration
Level 4	Daily meeting (stand up meeting), user stories, frequent releases	Self organizing team, 40 h week	Simplicity (simple design), on--site customer
Level 5	TDD, pair programming	Focus on continuous improvement	–

Currently, we do not know why Agile practices are abandoned and how this impacts the overall success of Agile implementations. Without such information, we are unable to evaluate the suitability of AMMs in industry. As the first step towards evaluating the suitability of the AMMs is to better understand the usage and retention of Agile practices, and identify the rationales for abandoning Agile practices.

In this study, we aim to identify the rate of usage of Agile practices, their retention, and the rationales for their abandonment. To achieve our aim, we conducted a web survey and 11 interviews with industry practitioners with experience in Agile.

The remainder of the paper is structured as follows: Section 2 presents related work. Section 3 presents the research questions and survey design. Section 4 presents the results and analysis of the survey. Section 5 discusses the results and Section 6 summarizes and concludes the paper.

2. Background and related work

2.1. Background

According Schweigert et al. [7], there are approximately 40 AMMs proposed by academia and industry consultants. Many AMMs usually associate a number of Agile practices with a maturity level [7, 8]. Practices are introduced gradually. As a team or organization becomes more mature, more Agile practices are adopted [8]. An overview over three typical AMMs is provided in Table 1.

The idea of adding more Agile practices as a team or organizations becomes more mature seems contradictory to current empirical studies that show that Agile practices like *TDD*, *pair-programming*, and *continuous integration* are abandoned [3]. This raises a question regarding the suitability of AMMs for industry, particularly when the AMMs do not provide rationales for the mapping of Agile practices to maturity models. Critics of the AMMs indicate that the AMMs are not fit for industry use [9] and that their recommendations are contradictory [8, 10]. In this study, we aim to evaluate the suitability of AMMs by investigating the usage and abandonment of Agile practices in industry through a survey and a series of interviews.

2.2. Related work

Kurapati et al. [16] performed a survey to identify commonly used Agile practices at project and organization levels. Their results show that the most commonly used practices both at project

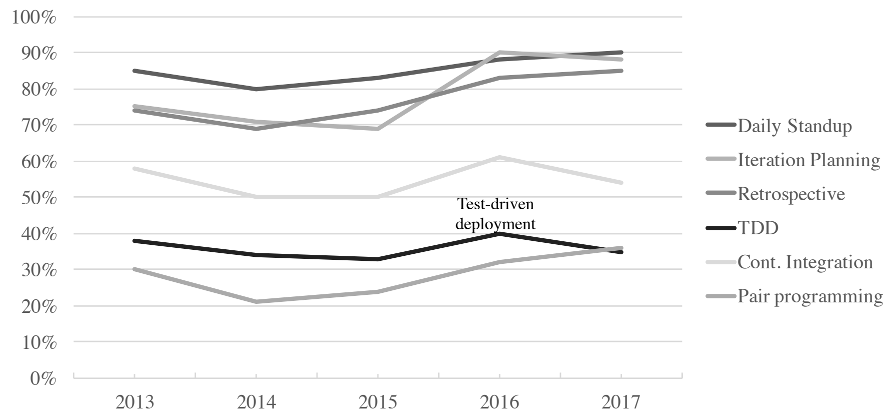


Figure 1. Fluctuation of Agile practice usage from Version One's State of Agile 2013–2017 [11–15]. The practices shown in the figure are for exemplification and the ones consistently reported across the annual surveys

and organization levels include *stand-up meeting, sprint and iteration, collective ownership*, and *tracking progress*. Less common practices both at project and organization levels include *simple design, TDD, pair-programming*, and *planning game*. One practice that is rarely practiced both at project and organization levels is *metaphor*. It is also interesting to highlight that the use of *metaphors* reported by Kurapati et al. turns out differently from Murphy et al. [17].

Kropp et al. [18] conducted a survey as part of Swiss Agile Study 2014. They distinguished three types of practices: technical, collaborative, and advanced practices. Technical practices include *refactoring*, *TDD*, and *coding standards*. Collaborative practices include *on-site customer, daily stand-up*, and *pair programming*. Advanced practices are *kanban pull-system, acceptance TDD*, and *Behaviour Driven Development (BDD)*. Their results show that more experienced practitioners implement considerably more practices compared to less experienced ones. Furthermore, less experienced practitioners implement primarily technical practices, meanwhile more experienced ones implement more collaborative practices. It is worth noting in this study *metaphors* is not included in the survey, unlike the previous survey by Kurapati et al. [16].

The two surveys described above, i.e. [16] and [18], report the results of Agile practice usage from one single calendar year. They do not capture whether the practices are continuously used or not.

Murphy et al. [17] reported results of five annual surveys internal to Microsoft over the course of six years. Their results show that practices like *code reviews*, *metaphors*, and *retrospective* are increasing in their adoption. Meanwhile, certain practices like *unit testing*, *TDD* and *pair programming* are decreasing in their adoption [17, Figure 4, p. 79].

VersionOne also conducts annual state of Agile surveys. We took the results from the annual survey over the past five years (2013–2017) and created a figure that presents the trend of the usage of some Agile practice [11–15] in Figure 1. The results of the annual surveys indicate that the use of Agile practices is fluctuating over the past five years, see Figure 1.

The surveys reported by Murphy et al. [17] and Version One [11–15] capture the increase and decrease of Agile practices usage over the years. However, the increase of some Agile practices from one year to the next does not indicate that those practices are being added, as suggested by AMMs. The decrease of some Agile practices does not indicate that those practices are being abandoned. It is possible that the respondents of the surveys from one year to the next are different. In the case of Murphy et al. [17] respondents who participated in one survey were not allowed to participate in the next survey. These surveys do not reflect the use of Agile practices in one context/team over time. Thus, the results cannot be used to assess the suitability of AMMs.

Solinski and Petersen [3] surveyed Agile practice adoption scenarios over time as practitioners transition from plan-driven development towards Agile. The survey identified Agile adoption scenarios which include an incremental adoption of practices, big-bang adoption – where plan driven practices are discarded and replaced by Agile practices, and complex tailored adoption processes. Their results also revealed that practices like *TDD* and *continuous integration* are being abandoned. However, their study did not focus on rationales for abandoning practices.

Indications of Agile practice abandonment is also reported by Ralph and Shportun [19]. Their case study revealed the abandonment of Scrum in distributed teams. One of the main factors associated with abandoning Scrum is the degradation of Scrum practices. Three Scrum practices that were difficult to implement due to distribution are *daily stand-up meeting, tracking progress using burn-down chart*, and *fixing sprint backlog.*

To summarize, current research indicates that some Agile practices are abandoned. However, current surveys have not yet focused on the rationales for abandoning Agile practices, or the time-frames from practice adoption to abandonment. Currently, we do not know how abandoning practices may influence the perceived overall success of implementing Agile methods. In this paper, we investigate why Agile practices are abandoned and whether or how this influences perceived success.

3. Research methodology

In this study, we aim to identify which Agile practices are being used and abandoned in the industry and the rationales for abandoning a practice to better understand practice adoption and the relevance of Agile maturity models.

RQ1. What is the rate of usage of Agile practices?

RQ2. Which Agile practices have been abandoned?

- RQ2.1. How long are practices in use before they are abandoned?
- RQ2.2. What are the rationales for abandoning these practices?

RQ3: What is the perceived success rate of Agile practices implementation?

- RQ3.1. Does the perceived success rate differ between respondents who retain practices versus respondents who abandon practices?
- RQ3.2. What are the used measures of success?

By "use" or "usage", we mean that an Agile practice is used or was in use at some point in time, while "abandoned" means that an Agile practice was used in the past, but is no longer used. To answer the research questions above, we conducted a survey and a series of follow-up interviews.

3.1. Survey

3.1.1. Sampling strategy

We distributed the survey to personal contacts and well-established professional groups in Agile software development on LinkedIn and Google Groups, i.e. convenience sampling. Distributing surveys over professional groups is a known way to distribute surveys as reported in [3, 16]. When using convenience sampling, which is a common strategy in software engineering surveys, it is important to describe the sample [20]. Following the guidelines from Linåker et al. [20], we define our sample as follows:

– Target audience: software industry practitioners who have experience in Agile practices adoption. Particularly, those who have experience in observing or experiencing when a practice is adopted and/or abandoned. In this survey, all practitioners from different industry domains, organization size and different levels of experienced are welcome to participate. However, this does not necessarily mean all responses will be considered (see Data Screening in Section 3.3).

– Unit of analysis: Agile practices which have been adopted and abandoned, their rationales, and perceived success rates.

– Source of sampling: professional groups or communities focused on Agile software development. Personal contacts who are known to work with Agile software development.

3.1.2. Survey design

We followed the recommendations from Robson [21] in designing a self-administered web-based survey. The survey was developed using the tool SoSci Survey (https://www.soscisurvey.de).

We included interactive sliders as a visual aid to allow respondents to indicate the start and/or end of Agile practice usage, see Figure 2. The survey design is adapted from Solinski and Petersen [3], who also investigated time-frames of Agile practice usage.

Similar to past surveys, we included a selection of Agile practice. However, there is no commonly agreed set of Agile practices. Different surveys include different sets of Agile practices. For example, Rodriguez et al. [22] include 16 practices; Kurapati et al. [16] include 25 practices. In this survey, we adopted the list used by Solinski and Petersen [3], which includes 7 plan-driven practices and 14 Agile practices. We chose this list because their survey is quite recent and comprises a manageable number of practices. In their survey, Solinski and Petersen [3], merged some practices, such as *short iterations and frequent releases*. We also merged two practices, if the practices are closely related. To see if two practices are related, we cross referenced the definitions of Agile practices described by Petersen [23] and Williams [2]. However, we separated Solinski and Petersen's combined practice "technical excellence" into its original sub-practices *refactoring*, *simple design*, and *coding standards*.

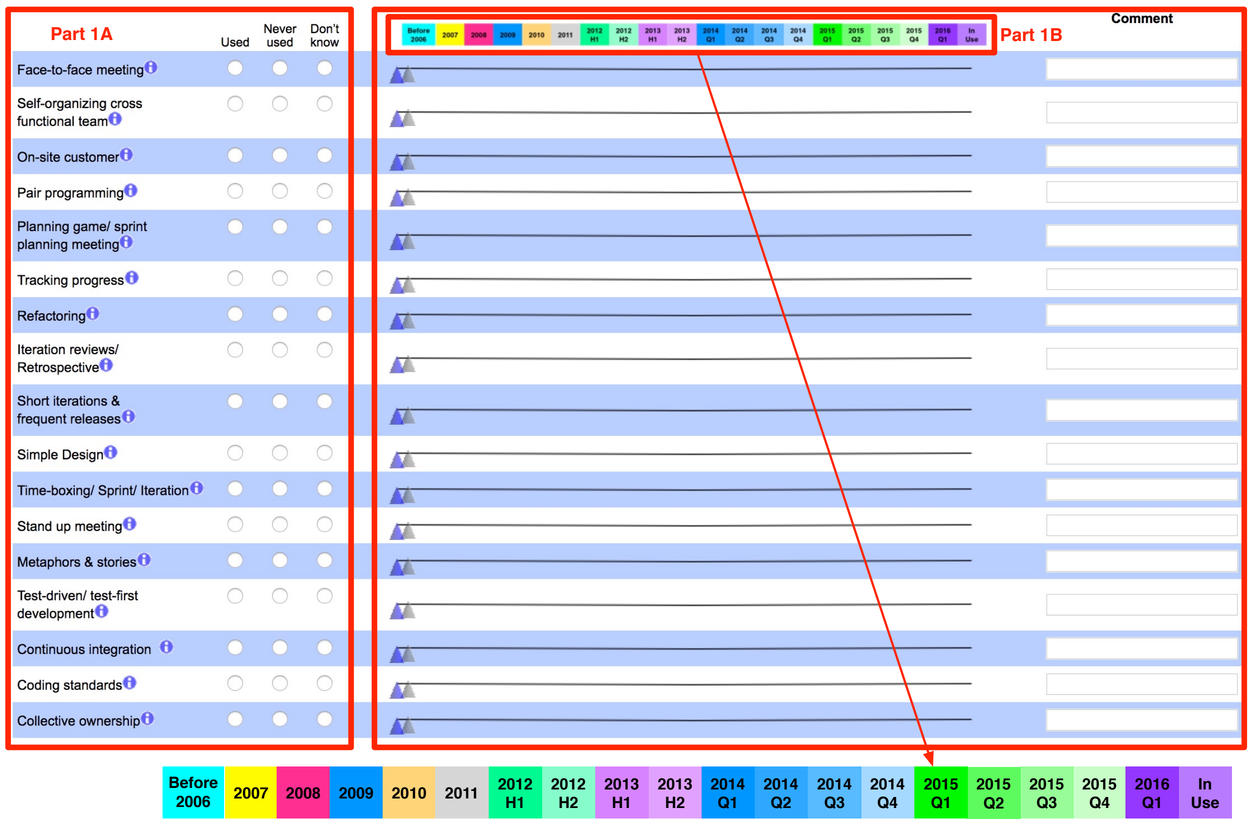
At the beginning of the survey, we briefly described the aim of the survey to the respondent. To avoid bias, we did not mention that we are looking for practices which had been abandoned. We described that we are interested in understanding the order in introducing Agile practices. The survey itself comprises five main parts. The detailed survey questions are available in Appendix B.

Part 1A. Agile practice adoption (RQ1). Respondents could indicate practice usage as "used", "never used", or "don't know". See Figure 2 Part 1A (to the left). Definitions of practices are available by hovering the mouse over the information icon. The practices included in the survey and their definitions can be seen in Appendix A. In this survey, we did not inquire which Agile framework, e.g. Scrum, eXtreme Programming (XP), etc. was used. This was done to avoid confusion from the respondents because it is possible that practitioners combine practices from different frameworks or on occasions also include plan-driven or waterfall practices [3, 24].

Part 1B. Start and end of Agile practice (RQ2 and RQ2.1). Using interactive sliders, respondents could indicate the start- and stop-time for when a practice was in use as shown in Figure 2 part 1B. The time-frame for the sliders is between <2006 and "Still in Use". When respondents indicated "never used" or "don't know" in Part 1A, the sliders are disabled. We used the interactive sliders to identify abandoned practices, so we did not bias respondents by explicitly asking for abandoned practices. Respondents could also leave optional comments or additional information regarding a practice.

Part 2. Perception and measures of success (RQ3). From Part 1B, we would be able to see which Agile practices were used, retained, and abandoned. The usage, retention, and abandonment of Agile practices represent a strategy for *Agile practice adoption.* We inquired the impacts of Agile practice adoption, as described in Part 1B, in terms of perceived success rate. *Success rate* is respondents' perceived degree of success of Agile practice adoption on their projects or teams. A Likert-type scale was used to indicate success rate, from very unsuccessful (1) to very successful (5). Respondents could also answer "don't know". Furthermore, we asked respondents to indicate how success was measured. We believe it is important to inquire what measures are used to indicate success, because different practitioners from different contexts may have different perceptions of success.

Part 3. Limitations and rationales (RQ2.2). We asked which challenges and limitations respondents experienced during Agile practice adoption according to Part 1B and, in particular, why practices were discontinued (if any).


Part 1A

	Used	Never used	Don't know
Face-to-face meeting	○	○	○
Self-organizing cross functional team	○	○	○
On-site customer	○	○	○
Pair programming	○	○	○
Planning game/ sprint planning meeting	○	○	○
Tracking progress	○	○	○
Refactoring	○	○	○
Iteration reviews/ Retrospective	○	○	○
Short iterations & frequent releases	○	○	○
Simple Design	○	○	○
Time-boxing/ Sprint/ Iteration	○	○	○
Stand up meeting	○	○	○
Metaphors & stories	○	○	○
Test-driven/ test-first development	○	○	○
Continuous integration	○	○	○
Coding standards	○	○	○
Collective ownership	○	○	○

Figure 2. Interactive sliders

Part 4. Contexts. We asked respondents to provide information about their personal background and organizational context: (1) their role(s), (2) years of experience, (3) number of team members involved in software development, (4) team-setting (collocated or distributed), (5) how Agile practice adoption was decided (team-level or company), (6) industry domain(s), and (7) type(s) of software systems being developed (classification is adopted from [25]).

Part 5. Contact. We also asked the respondents to provide their names and email addresses, for follow-up interviews or to receive a copy of the survey results.

3.1.3. Survey pilot and execution

The sliders made the survey more complex and increased the risk that questions are not well understood. To mitigate these risks, we piloted the survey with five colleagues of the authors and five industry practitioners with experience in Agile software development.

Regarding the pilot, some industry practitioners felt that the definitions of some Agile practices were too specific and might not be applicable in their contexts. To address this issue, we reformulated the definitions. Two pilot respondents had difficulties to move the sliders. We resolved this problem by adding instructions on how to use the sliders. After addressing the feedback from the pilot, we deployed the survey, which was open between March–July 2016.

3.2. Interviews

3.2.1. Interviewees recruitment

Interviewees were recruited from the survey respondents who left contact information for further inquiries. Twelve invitations were sent out, and three confirmed for follow up interviews. We then recruited eight additional interviewees through personal industry contact and referrals. For the new recruits, we also asked them to fill in the survey prior to the interview to maintain consistency and helped us to formulate interview questions.

In total, we had 11 interviewees. Our interviewees represent a wide variety of contexts. They came from various industry domains and geographic locations. More details on the interviewees can be found in Table 2.

3.2.2. Interview design

The goal of the interviews was to gather richer and better contextual information about the use and abandonment of Agile practices. In the interviews, we used semi-structured interviews. The interviews were done face-to-face whenever possible. Otherwise, the interviews were done over the telephone or video call. Prior to the interview, we sent each interviewee a summary of their answers from the survey. Each interview lasted 45–60 minutes and was recorded and transcribed. In the interviews, we inquired the following:

- Interviewee's roles and responsibilities, short description of the product being developed;
- The interviewee's survey answers were revisited and further clarified:
  - Why did you mark (enumerate Agile practice marked as "never used") as never used? (RQ1),
  - Why did you mark (enumerate Agile practice marked as "don't know") as don't know? (RQ1),
  - Could you please elaborate the reasons for abandoning (enumerate Agile practices which were no longer used from Part 1B)? (RQ2.2);
- Wrap-up. Inquire the interviewee's impressions on the interview.

3.3. Data analysis

Data screening. Prior to the analysis of the survey data, we carefully scrutinize each dataset to ensure their reliability. We checked each respondent's answers to each question. For example, we cross referenced the participants' experience (in years) and the time frame indicated in the sliders. We also checked the respondents' answers to the open-ended questions. We excluded a response if a respondent did not provide a comprehensible answer to one of the open-ended questions. We also excluded a response if a respondent indicated that most or all of the practices had been abandoned and did not specify that it was past experience.

Rate of Agile practice usage (RQ1). We used descriptive statistics to analyze the rate of Agile practice usage, i.e. practices that are marked as "used" by the survey respondents.

Agile practice abandonment (RQ2). For all practices that were indicated as "Used" (in Part 1A of the survey), we checked the slider position for "practice end". If this position did not indicate "In Use", we considered the practice as abandoned and calculated the timespan of use by means of the slider positions for start and end of use, respectively. We also calculated the *abandonment ratio* for each Agile practice to calculate the proportion of the number of times a practice is abandoned to the number of times a practice is used. To answer RQ2, we also included the results from the interviews. To analyze the interview transcripts, we used f4analyse tool (https://www.audiotranskription.de/english/f4-analyse) to help with coding steps. First, we performed line-by-line coding as an approach to open coding [26] on the interview transcripts. Open coding was followed by focused coding to identify common themes from the data. The result of focused coding can be seen, for example, in Table 5. The coding process was primarily done by the first author. To minimize bias, another co-author conducted post-hoc validation on the coding done by the first author.

Success rates and measures (RQ3). We analyzed the success rates of adopting Agile practices across domains, and retain vs. abandoned. We also used descriptive statistics to analyze the success rates. To cross-tabulate the industry domain and the success rates, we used the "Crosstab" feature in SPSS. To identify the measures of success from the survey, we employed qualitative coding similar to the one used for analyzing the interview data. First, we tabulated all responses to each relevant question using a spreadsheet and f4analyse tool. We then used open coding [26] to assign codes to text fragments. For example,

Table 2. List of interviewees and their contexts

ID[a]	Location	Role	Experience	Team size[b]	Market	Domain	Context overview
R11	Indonesia	Project Manager	6 years	100	Market driven, internal use	Insurance	IT Department of a multinational Fortune 500 company. Adopted 13 practices except pair-programming, TDD, and metaphors & user stories.
R14	Brazil	Developer, Trainer, System architect	3 years	20	Internal use	Government (Court)	IT Department from the Brazilian court of accounts. Adopted 15 practices except pair programming and retrospective. Abandoned on-site customer and tracking progress.
R32	Canada	Developer, Quality Assurance, System Analyst	6 years	13	Market driven, bespoke	Independent Software Vendor (ISV)	Start-up company initiated in 2012. Adopted 14 practices except for on-site customer, simple design, and TDD. Abandoned pair programming and tracking progress.
R33	Sweden	Scrum Master	6 years	6	Internal use	Telecoms	A small project team within a large multinational company. Adopted 15 practices except on-site customer and TDD. Abandoned 13 practices except face-to-face meeting and stand up meeting.
R34	Indonesia	CEO	3 years	33	Bespoke	ISV	Start-up company initiated in 2014. Adopted 14 practices except TDD, collective ownership, and metaphors & user stories.
R35	Ireland	Scrum Master, Developer	3 years	6	Bespoke, market driven, maintenance	ISV	Start-up company initiated in 2012. Adopted 14 practices, except TDD, coding standard, and simple design.
R36	Sweden	Program Manager	23 years	1000+	Market driven	Telecoms	A solution development program in a large multinational company. Adopted 16 practices except TDD.
R37	Sweden	Scrum Master	20 years	1000+	Market driven	Telecoms	A solution development program in a large multinational company. Adopted 16 practices except TDD.
R38	Sweden	Scrum Master, QA	7 years	70	Market driven	ISV	A project in a large multinational company. Adopted 15 practices except on-site customer and simple design.
R39	USA	Researcher, Developer	3 years	6	Bespoke, market driven	Research & development, biomedical	A project in a university to develop biomedical research support tool. Adopted 12 practices except pair programming, tracking progress, stand up meeting, metaphors & user stories, and TDD.
R40	Finland	CTO, Developer, Scrum Master	6 years	11	Market driven	ISV	A start-up company initiated in 2012. Adopted 10 practices, except on-site customer, planning game, refactoring, retrospective, metaphors, TDD, and collective ownership.

[a] Respondent ID according to the order they are received in the survey tool.
[b] Reflects the size of software development team affected by the Agile implementation. Not overall company size.

for the following response regarding used success measures: “Success can be measured by completion of tasks on time with high quality and without any blockers”, we assigned two codes: *time to deliver* and *product quality* as measures of success. The measures of success were then classified into product, process and resource measures according to Fenton and Bieman [27].

3.4. Validity threats

In reporting the validity threats, we follow the classifications suggested by Petersen and Gencel [28].

Theoretical validity. It refers to pertains to the issue of capturing the construct intended to be collected. Both the survey and the interviews, are retrospective. The respondents may not remember precisely when an Agile practice was introduced. To minimize the issue, we did not inquire exact months or dates for the start or the end of a practice. We only refer to the year, half a year, or quarters. The slider design does not support exact dates and only one start and one end time. To minimize the issue we added comment text boxes next to the sliders to supply details. Maturation could pose as a threat if the survey takes too much time to complete. To minimize maturation, we minimized the number of included Agile practice, i.e. 17 practices. It is possible that we missed one or more Agile practices. To reduce maturation, we merged practices that are similar in their definitions, as described in [23]. It is also possible that merging some of the practices caused confusions to the respondents. In this survey, we also provided definitions of the Agile practices primarily from the literature, e.g. [2, 23]. It is known that how Agile practices are implemented in the industry may differ from their definitions in the literature [2]. This may lead to respondents answering “don’t know” or “never used”, when the practices are actually in use. These issued are partially mitigated by piloting the survey and performing follow up interviews with 11 of the survey respondents. Another concern pertaining theoretical validity is with the sampling. In this survey, we used convenience sampling by recruiting participants from professional groups and personal contacts. The former may lead to reliability issue, while the latter may lead to bias. To minimize reliability issue, we checked each response to ensure coherence (see *Data Screening* Section 3.3). For example, if a respondent indicated to have 1–3 years of experience, but used the sliders indicating a period longer than that, we deemed the answer to be invalid. To minimize bias from the personal contacts, as well as the other respondents, we did not specify that we aim to collect Agile practices that are being abandoned. It is also important to clarify that these personal contacts were not individuals whom the authors had prior close collaborations. Thus, they were never given information about the plan of the study.

Descriptive validity. It concerns with the accuracy of capturing the reality. In this study data collection was done through a survey and interviews. As researchers, we cannot observe the reality, and the responses we obtained are based on the respondents’ perception. For example, a respondent’s experience can influence his/her answers; a new hire may not be aware that a practice was used before but has been abandoned. It is also possible, that a respondent perceives a practice was used because he/she used it, but it was not institutionalized in the team or project. The follow-up interviews helped to capture better information that was otherwise missing from the survey. However, in survey and interviews studies, such a threat cannot be fully eliminated, since no actual observation was done. Although we were not able to eliminate the issue, it is important that we acknowledge it. In this survey we provided instructions for the respondents to reflect on an experience that they were most familiar with, it could be an experience from a specific team or a specific project. The experience could also be from present or past experience. It is possible that a respondent reflected on past experience, and indicated all practices had been abandoned. For such a case, unless the respondent wrote a note that it was past experience, we deemed the answer to be invalid.

To improve thoroughness and trustworthiness of the survey, we reported as many details as possible regarding the design and execution of the survey, following the criteria described by Stavru [29]. A self-assessment on the thoroughness of our survey using Stavru's criteria and calculation procedure resulted in a score of 0.8 on a scale 0–1 (see Table C.1 in Appendix C for details). Stavru does not provide interpretation of the scale. However, our score is higher than other Agile surveys examined by Stavru in [29], where the highest score was 0.64. This indicates that we have provided sufficient information to demonstrate the thoroughness of our survey [29].

Interpretative validity. It concerns with researchers' bias in drawing a conclusion. This study primarily relies on qualitative data collected from a survey and from interviews. Researchers bias can affect the conclusions that are drawn. In analyzing the data, the first author was responsible for the qualitative coding. To reduce bias, another co-author validated the coding post-hoc after the first five interviews, to see if there could be disagreements in the codes.

Generalizability. It refers to the extent that the results of the study are generalizable to a larger population. In this study, both for the survey and the interviews, we used convenience sampling. The selection of the respondents was non-purposeful and based on willingness. Respondents have various roles and tasks in different organizational contexts. However, some roles such as consultant and C-level managers are under-represented. Furthermore, most of the respondents work in small organizations. Although we did not collect company name and geographical location of the respondents, we could ascertain that our sample represents 20 unique organization from 11 different countries. Although some countries like the Canada, Italy, and New Zealand are under-represented, our sample represents different geographical locations. In this survey, we also small sample size. We cannot claim that our results are generalizable to a large population or in anyway represents the current state of Agile practice. However, the demographics of our respondents include a large variety of contexts that adds to the richness of the data and minimizes the risk of confounding factors that could be present due to a homogeneous context.

4. Results and analysis

In total, 200 people started the survey, 70 completed the survey but only 43 answers were valid, i.e. consistently answered part 1–4 of the survey. Out of 43 respondents, 32 of them completed part 1A and used the sliders from part 1B of the survey. The remaining 11 respondents did not use the sliders (part 1B). Including the new interviewee recruits, in total, we have 51 respondents and 40 of them used the sliders. From 40 respondents who used the sliders, 22 retained all practices that were used. Meanwhile 18 abandoned one or more practices.

Out of 51 respondents, 10 participated through direct invitations, and 3 participated through referrals. In the survey, we did not inquire company name and location where the respondents were or had been employed. Based on direct invitations, referrals, and a number of respondents who provided their work emails, we could ascertain 20 unique companies from 22 respondents. We could also ascertain the geographic location of 19 respondents; they were from Sweden (5), Ireland (3), US (2), Indonesia (2), Canada, New Zealand, Finland, Portugal, Brazil, Germany, and Italy (1 of each).

The 51 respondents were primarily developers (20; 39.2%) followed by Scrum Masters (15; 29.4%) and quality assurance specialists (13; 25.5%). Please note that multiple roles could be selected. Further roles are system architect and department head (8; 15.7% for each), project manager and department head (each 7; 13.7%), business analyst, system analyst trainer, product owner, C-level managers (e.g. Chief Executive Officer, Chief Technical Officer, etc.), and other roles (<6; <10%).

Regarding their level of experience in software development, 14 (27.4%) respondents had more than 6 years of experience, 15 (29.4%) had 3–6 years of experience, 15 (29.4%) had 1–3 years of experience, and 7 (13.7%) had less than one year of experience. Most of the respondents (21; 52.5%) were part of a small organization with less than 50 people involved with software development. Eleven (27.5%) were part of organizations with 50–249 people, and 7 (17.5%) were part of organizations with 250–4999 people.

In terms of distribution, 28 (54.9%) of the respondents mentioned that their Agile software development teams were collocated; 19 (45%) of them worked in a single team and nine (17.6%) in multiple teams. The remaining 23 (45.1%) stated that their Agile software development teams were distributed; 10 (19.6%) of them worked in a single distributed team and 13 (25.5%) in multiple distributed teams.

Regarding application domains (multiple selections possible), most of the respondents were from independent software vendors (17; 33.3%), followed by financial services (15; 20%). Respondents also came from the following domains: research and development (11; 21.6%), telecoms (12; 23.5%), medical (8; 15.7%), media and entertainment (4; 7.8%), government (3; 5.9%), and manufacturing (1; 2%). For the types of software systems that respondents develop and type of market, please refer to Figure 3a and 3b.

To complement the survey, we also conducted 11 interviews with industry practitioners. The list of interviewees and their respective contexts are presented in Table 2.

4.1. Usage of Agile practices (RQ1)

Figure 4 shows the rate of Agile practice usage. From Figure 4, we can see that out of 51 respondents, *face-to-face meeting* was the most commonly used Agile practice among our respondents (48 respondents), followed by *tracking progress* (47 respondents). Other commonly used Agile practices by our respondents were: *self-organizing team*, *planning game*, and *retrospective*. Practices like *TDD* (27 respondents) and *pair-programming* (28 respondents) were less commonly used by our respondents.

The follow up interviews identified Agile practices that were not included in the survey, they are: (1) Behavior-driven development/BDD (R35), (2) Scrum of scrums (R38). R32 mentioned that in addition to *retrospective* at the end of a sprint, they also do a project level retrospective which was done every two months.

The follow up interviews also revealed that some respondents interpreted the definition of Agile practices slightly different to our definitions. R35 and R36 indicated in the survey that *on-site customer* was used. However, in the follow up interviews, they clarified that they did not actually have customers present on their premises. Rather they have a dedicated team member who acted as a proxy to the customers, i.e. *product owner*.

The rationales for never using certain Agile practices are summarized in Table 3. From the interviews, we identified that respondents R11, R14, R32, R39, and R40, marked some of the practices as "never used" or marked as "don't know" because they were not adopted according to our provided definitions or were not adopted consistently. For example, when inquired why *stand up meeting* was never used respondent R39 mentioned that " because of the word daily in the definition, we do not do daily meeting". Meanwhile respondent R11 mentioned the reason for marking "do not know' for *collective ownership* is because the project involved outsourced developers and the level of collective ownership varies from the internal team to the outsource team: "internal [team] is not a problem, but the outsource team has no collective ownership". This indicates that the usage of Agile practices is not binary (used or not). Often Agile practices are modified from how they are defined or implemented inconsistently.

From Table 3, we can see that some practices are not suitable in certain contexts. Some practices may not be applicable given certain contextual factors like regulation, team/organization's culture, and organization set-up. The characteristics of the software system, e.g. legacy code and product complexity, could also make some Agile practices unsuitable.

4.2. Abandonment of Agile practices (RQ2)

As mentioned earlier, 18 of the respondents abandoned one or more Agile practices. Each respondent abandoned at least one Agile practice. One of the 18 respondents abandoned up to 13 Agile

Table 3. Rationales for never using certain Agile practices and the supporting quotes from the respondents

Rationale	Practice	Quotes (with Respondent's ID)
Incompatibility with the domain or market of development	Short iteration	Release of each sprint to end customer is not possible in case of regulatory development (R20)[a].
	On-site customer	Our customers are 100M people (R13)[a].
		We are product company, it is a [software as a service] product over the Internet (R40). Some of our customers are not even in the province (R32).
Challenges in implementing a practice	User stories	The biggest challenge was conforming to the structure of developing user stories (R19)[a].
	TDD	We do not have the patience to follow through with it. It is quite challenging with a big ecosystem [of 26 products] like this (R36).
	Metaphors	We use user stories but not metaphors, metaphors are too obscure for most people to grasp (R35).
Product complexity	Simple design	The product we were working on was extremely complex, we had a lot moving pieces and that was an unavoidable complexity the domain was complex [. . .] the hardware aspect definitely have to do with it, hardware and firmware development (R32).
	TDD	Our product is very explorative, [we are] creating new software, we rather implement TDD next time (R40).
Legacy code	Simple design	We are left with a mess from the previous development team. We are adding and maintaining the legacy we are left with to get the product to the market (R35).
	TDD	We have a lot of [legacy] in our code, [it was not easy] for us just to jump into [TDD] [because] the old code was not done in that way (R37).
Organization set up	On-site customer	we never interact with customers because were in the R&D department, the department that interacts with the customers is called customer unit (R33).
Lack of resources	Collective ownership	We have a massive product and too few people, collective ownership is not possible, we need specialists (R40).
Lack of management involvement or enforcement	Retrospective	Most of time management would trust the team to work, they [would not] be picky and asking people to do retrospective and that kind of thing (R14).
	TDD	I [do not] know why we [do not] use TDD, We at [the company] just never use TDD (R33).
	PP	Management [did not] talk about it at all. I [do not] think we ever discussed whether to use pair programming or not (R14).
Lack of perceived value	Planning game	There is no need for a planning game because each developer is responsible for a component of a feature. I [do not] think planning game helps in this case. Just keep releases small and often (R40).
	Refactoring	It [does not] make sense to refactor because the components that you refactor would be obsolete anyway in a very short time (R40).
Conflict with team's culture	Retrospective	We want to foster the kind of culture where you are not keeping something for a [sprint]. You just bring it up immediately (R40).

[a] Respondent provided answer through the survey.

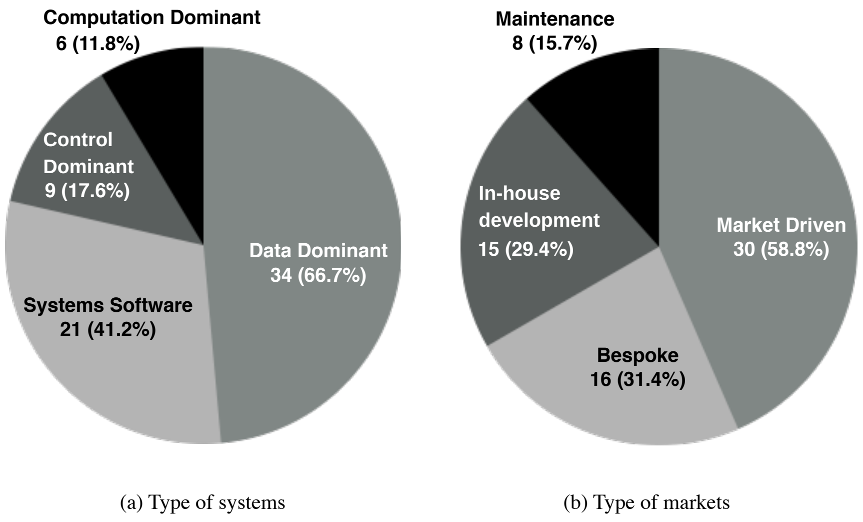


Figure 3. Respondents' type of system and type of markets (multiple selection possible)

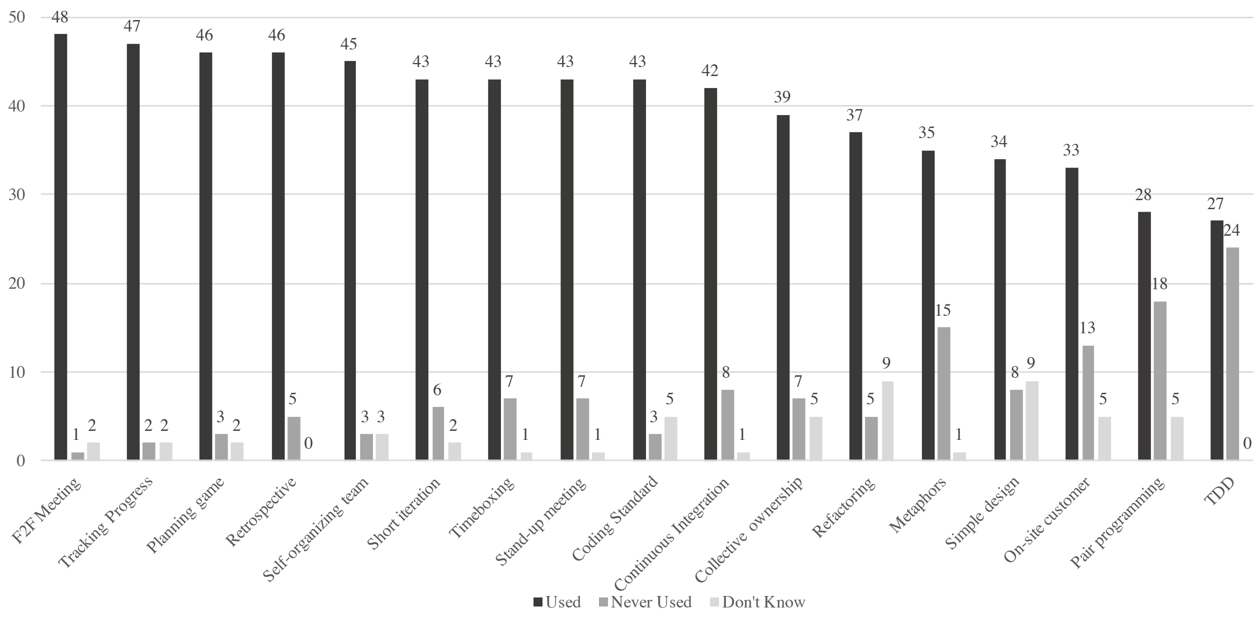


Figure 4. Adoption of Agile practices

practices. All 17 Agile practices included in the survey were abandoned at some point. From Table 4, we can see that *face-to-face meeting* has the lowest abandonment ratio (0.05). Meanwhile, *Tracking progress* has the highest abandonment ration (0.29) followed by *planning game* (0.2). This finding may indicate that certain practices, like *face-to-face meeting*, are more easily retained than others. Meanwhile, a practice like *TDD* may not be as popular, but once it was adopted, it is more likely to be retained, as we can see the abandonment ratio is quite low (0.11).

From Table 4, the number of respondents who abandoned individual Agile practices is relatively low when compared to the number of respondents who retained the practices. This shows that most of the time each Agile practice is still in use.

4.2.1. Usage until abandonment (RQ2.1)

Table 4 summarizes the periods of time that an Agile practice was in use. Practices are most often abandoned within the first half year after their introduction. After 3 years of use, the rate

Table 4. Agile practices that have been abandoned and how long they had been in use before abandonment

Practices	≤6 mon	≤12 mon	≤ 24 mon	≤36 mon	≤48 mon	60+ mon	Abandon	Still in use	Total usage[a]	Abandon ratio[b]
Tracking progress	3	2	3	1	1[c]	1	11	26	37	**0.29**
Planning game	3	1	1	1		1[c,d]	7	28	35	0.2
Retrospective	2	1	2	1			6	30	36	0.17
Time-boxing	2	1	1	1			5	30	35	0.14
Collective ownership	2		1	1		1[c,d]	5	25	30	0.17
Self organizing team	1	1	1	1			4	34	38	0.11
Pair programming	2		1	1			4	17	21	0.19
Simple design		2		2			4	22	26	0.15
Stand up meeting		2	1				3	32	35	0.08
Refactoring		2		1			3	26	29	0.1
Short iteration	2			1			3	29	32	0.09
Metaphors and stories	1	1		1			3	24	27	0.11
Continuous Integration			1	2			3	30	33	0.09
TDD	1	1					2	15	17	0.11
F2F Meeting	1				1		2	37	39	**0.05**
On-site customer	2						2	22	24	0.08
Coding standard	1			1			2	32	33	0.06
	23	13	12	15	2	2				

[a] Total usage based on 40 respondents who used the sliders. [b] Ratio = abandon/total usage.
[c] Respondents in financial domains. [d] Response from the same respondent.

of abandonment drops significantly. Only *tracking progress, planning game, collective ownership* and *face-to-face meeting* were abandoned after having been in use for more than 3 years.

This finding may indicate that in some contexts, certain practices are not suitable to be introduced in the first place, or introduced in the wrong order due to dependencies on other Agile practices. Also, as the findings from subsection 4.1 shows that Agile practices may be modified or implemented inconsistently, it is possible that the modifications, or the lack thereof, has undesired side effects that may present themselves at various time periods. The rationales for abandoning Agile practices are presented in the following subsection.

4.2.2. Rationales for abandonment (RQ2.2)

Eight respondents provided rationales for abandoning the following practices: *pair programming, tracking progress*, and *on-site customer*. Meanwhile, two respondents, R28 and R33, abandoned 5 and 13 practices respectively. They did not provide a rationale for each practice. Instead, they provide a common rationale for abandoning a group/set of Agile practices (indicated as Not specific in Table 5). Most rationales were obtained for *tracking progress*. Table 5 summarizes the rationales for abandoning Agile practices.

The statements from R14 and R38 in the discontinuation of *tracking progress* indicate that Agile practices dependencies are not well understood. In the case of R14, *tracking progress* was introduced before sprint planning was established. Because sprint planning was not done, new tasks could be added throughout the week, and tracking progress became ineffective, as respondents R14 explained: “It seems like we were walking backwards. We were working towards the end of the week, and things just got worse. Because somebody would suddenly add a workload to the sprint.” Meanwhile, in the case of R38, tracking progress was introduced before the team members develop better product knowledge. This shows that there could be prerequisites before introducing certain Agile practice. The prerequisites could be other Agile practices or acquiring product or project-related knowledge.

From Table 5, one of the more interesting rationale for abandoning one or more Agile practices is the influence of a person, as reported by

Table 5. Rationales to abandon Agile practices and the supporting quotes from the respondents

Rationale	Practice	Quote (with Respondent's ID)
Poor estimation and team dependency	Tracking progress	Due to bad estimation and dependency on other teams we are unable to track progress by burn-down chart (R32)[a].
Lack of product knowledge	Tracking progress	[The team members] complain that we [do not] have the product knowledge, how do we estimate it if we [do not] know the complete technicalities (R38).
Team member discomfort	Pair programming	People were uncomfortable and people did not really want to engage in that (R32).
Lack of engagement	Tracking progress	Half of the were tracking progress and they other half [were not], management did not really care (R14).
Conflict with other Agile values	Pair programming	The idea of sustainable pace, [...] we are only expected to be at the office at certain core hours [...]. I would be one of the people showing up around 9.30-16.30 [...]. so if I want to pair program with one of the latecomers, it would only really work from 13–15 (R32).
Influence of a person	Not specific	It was because one person was quite very opinionated, the person thought why do all these things, it's a waste of time (R33)[b].
	On-site customer	The guy [who initiated on-site customers] went on vacation and he did not come back (R14).
	Tracking progress	The new product owner did not want/care for [statistics], and the team did not demand them (R32).
Lack of perceived values	Tracking progress	As we do product development of a rather mature product, the tracking of progress was not all that valuable. Stuff at the top of the backlog has most value. Stuff lower has a lower value, and will be released later. No real forecast of this was needed (R21)[a].
	Tracking progress	We just try to push things to production all the time (R40).
	Tracking progress	The team did not feel the need for it (R30)[a].
	Not specific	The part that can be handled by Agile is finished. Other part cannot use Agile (R28)[a,c].
Dependency on other practice	Tracking progress	We tried to do tracking progress but sprint planning was not done [yet] (R14).

[a] Respondent provided answer through the survey. [b] 13 out of 15 Agile practices were abandoned.
[c] 5 out of 12 Agile practices were abandoned.

R14, R32, and R33. Respondent R14 mentioned that *on-site customer* was adopted for only two months because the person in charge had to leave the company. This individual was crucial to make *on-site customer* worked smoothly because the person can bridge between the technical team and the end users (court officers): "He was both an engineer and a lawyer. So he could very easily talk to the business people and to us". Meanwhile, R33 indicated that the practices were adopted for up to three years until they are abandoned: "They've been practicing Scrum since 2012. Suddenly in 2015, they stopped completely [...]. They just dropped everything, and they only do stand up meeting [...]". This indicates the influence of an individual can affect the abandonment of Agile practices, but also how long they were adopted until abandonment.

From Table 5, we can see that there could be more than one cause to abandon an Agile practice. For example, we identified multiple reasons for abandoning *tracking progress*. One of the more common reasons is lack of perceived values. To abandon *tracking progress* due to the decrease of perceived value seems counter intuitive because the need for tracking progress would increase as the product grows and more tasks are associated with delivering the product.

In the case of tracking progress, it is possible that the practitioners did not completely abandon tracking progress altogether, but abandoned tracking progress according to the definition in the survey. Respondent R14, R23, R33, and R38 indicated in the survey and interviews that they use *Kanban board* to replace burn up or burn down charts as a means of tracking progress.

The results from the survey and follow up interviews indicate that there could be multiple factors that can contribute to abandoning an Agile practice. Engagement, knowledge, and dependencies between development teams can contribute to the abandonment of one or more Agile practices.

4.3. Perceived success of Agile practice adoption (RQ3)

In Figure 5, we looked at the perceptions of success of Agile practice adoption by industry domain to see whether our sample shows differences between domains.

From Figure 5, we can see that the adoption of Agile practices was generally perceived as being successful. Most of the respondents (30; 60%) perceived the adoption of Agile practices as successful and 11 (22%) as very successful. Only one respondent (2.8%) perceived the adoption of Agile practices in his/her organization to be unsuccessful. No respondent answered "Very unsuccessful". There were only minor differences between domains.

In the follow up interviews, we identified a number of factors that contribute to the perceived success:

- Management: Trust and commitment from managers on Agile adoption (R32), a clear vision of Agile transformation from the upper management (R37).
- Leadership: Ability of the leader to provide guidance (R38).
- Team members: Engagement (R36), experience and technical skills (R40).

During the follow up interview, R11 who indicated unsuccessful adoption of Agile practices mentioned that the issue was with the company policy, which is also related to management, of providing documentation at the end of every sprint: "if you want to be effective, with the small chunks of deliverables, there are more effort because the amount of procedure is the still the same as the big one. Agile implementation somehow is "heavier" on the procedure side. For every deliverable we need to provide documents like technical documentation, deployment guide, training material, [user acceptance test] sign off". Respondent R11 also felt that the kind of product they were developing did not fit Agile: "You need 6 months to develop the core engine. I cannot split a function into two releases, because it will be useless for the user. We have heavy rule engine and workflow. For this type of project, Agile does not work".

The respondents who perceived Agile practice adoption as very successful or successful (43 respondents) were primarily from small and medium sized organizations (25 and 15 respondents respectively out of the 43 respondents). This, however, does not indicate that Agile practice adoption is more successful in small organizations. We simply cannot make such assertions, since we have a small sample size and more than 50% of the respondents were from small organizations. Performing inferential statistics to examine the correlation between success rate and organization size would not be meaningful.

Overall, our survey respondents perceived their Agile practice adoption to be successful. We did not find significant variations of perceived success across the different domains. We identified factors that may influence the perception of success from the respondents, such as management, leadership, and team members.

4.3.1. Success rates: retained vs. abandoned practices (RQ3.1)

We also compared the success rates of 40 respondents who retained all adopted Agile practices and respondents who abandoned one or more Agile practices. From Figure 6, we can see that the perceived success of Agile practices was similar in both groups. This result indicates that

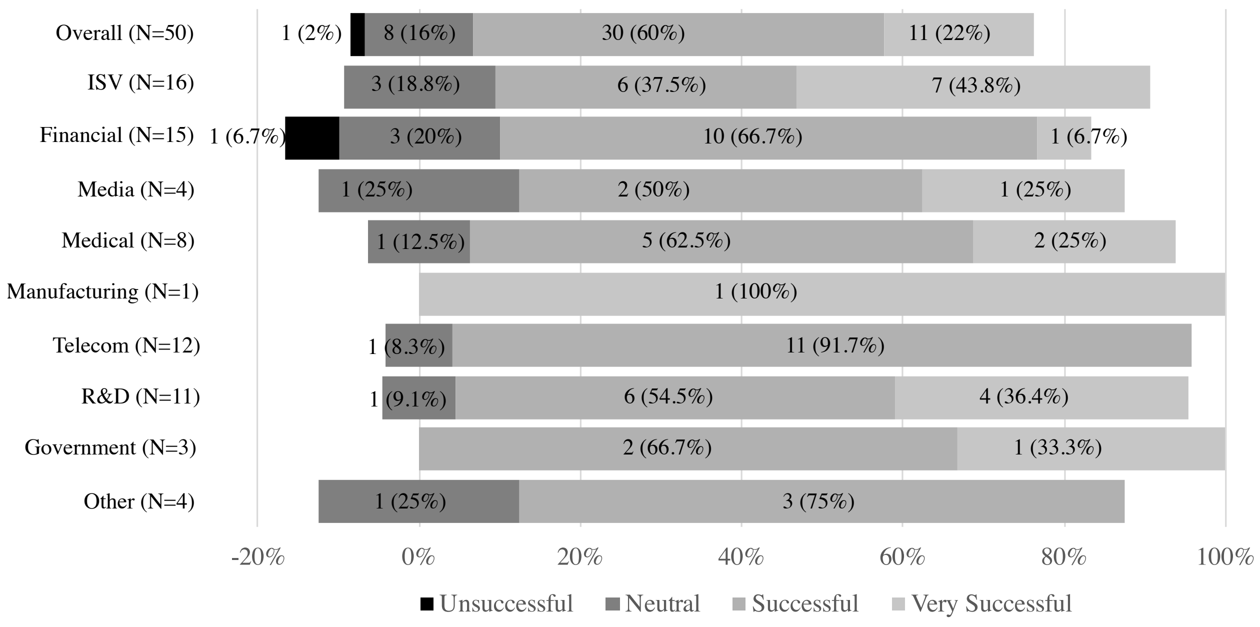


Figure 5. Perception of success of Agile practice adoption for all participants ("Overal", top row) and by industry domain (row 2–9). $N = 50$; one of the 51 respondents did not answer the question about perceived success

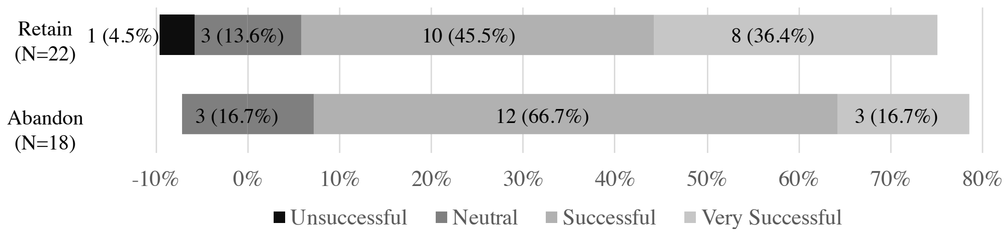


Figure 6. Success rates: retained versus abandoned practices

an abandoning of one or more Agile practices might be required to achieve or sustain an overall successful Agile adoption.

However, it is also important to remember that not all respondents adopted the same set of Agile practices. Those who achieved successful or very successful Agile adoption by retaining all Agile practices may have found the more suitable set of Agile practices or have successfully found an optimal way to tailor the Agile practices. We, however, do not claim that those who abandoned practices were less successful in selecting the suitable of Agile practices.

4.3.2. Measures of success (RQ3.2)

As we can see from subsection 4.3 and 4.3.1, our survey respondents generally perceived their Agile practice adoption to be successful. It is important to understand how success is measured since there could be different ways to perceive success. We collected measures of success from 35 respondents and classified them into product, process, and resource measures [27, Chapter 3, pp. 87–98]. Table 6 summarizes the measures that were reported the respondents and the number of respondents that reported them.

Among the product measures, "product quality" and "customer satisfaction" were named most frequently (12 and 9 times, respectively). Among the process measures, "time to deliver" was named most frequently (16 times). "Team spirit (happiness)" was the most frequently named resource measure.

Respondents considered a large diversity of indicators as being success-relevant, including measures from all three categories. Table 6 lists 16 unique "process" measures, 11 unique "resources" measures, and 8 unique "product" measures. This result shows that success of Agile practice adoption can be perceived in many different ways.

Looking at the number of different measures and the number of respondents who contributed them, we can see that our respondents put much focus on how well a "process" is executed and on the quality of the "product". On a more detailed level, the respondents focused on product quality, customer satisfaction, and time to deliver, and good team spirit. This result is in line with the overall goals of the Agile manifesto [30] and the principles behind it.

We can see that the respondents reported measures at different levels of granularity. For instance, most respondents referred to "product quality" or "customer satisfaction" as measures for quality without going into detail about how those were measured. Few respondents named actual specific measures, like "number of defects/bugs" or "number of met sprint goals".

When looking at the respondents' experience and roles, we could not identify any specific patterns regarding the measures they provided. Respondents with more technical roles, e.g. developers or testers as well as those with managerial roles provided both specific and generic measures.

5. Discussion

In this study, we conducted a survey and 11 interviews on Agile practices adoption and abandonment. To guide the discussion, we reflect our findings and compare them against known recommendations from Agile maturity models (AMMs).

The respondents of our survey indicate that face-to-face meeting and tracking progress are frequently used. Meanwhile, TDD and pair programming are less commonly used by our survey respondents. From the follow up interviews, we identified different rationales from our respondents why some Agile practices were never used. The rationales for never using certain Agile practices indicate that all Agile practices are not always applicable in different contexts. Agile practices are not used due to incompatibility with the development context, challenges, or lack of management enforcement. AMMs typically recommend to gradually add more and more Agile practices (see Table 1) without considerations on whether the practices are suitable within a context. For example, 24 of our respondents never used TDD, but two out three AMMs that we exemplified in this paper recommend that TDD is to be introduced. Our study also indicates that Agile practices could be modified from its definition. However, the AMMs that we exemplified in this paper do not provide their definitions of the Agile practices. This raises the question regarding the suitability of AMM in industry.

The result of our survey indicates that not all Agile practices are sustainable. Eighteen of the respondents have abandoned one or more Agile practices. Our survey also shows that Agile practices were more frequently abandoned within the first six months after their adoption. Meanwhile, some Agile practices, like *continuous integration*, *planning game*, and *collective ownership* were adopted for extended period of time. This finding complements the findings of a previous study by Solinski and Petersen [3]. The AMMs indicate that Agile practices are to be gradually added. However, in certain contexts, it is not always possible to sustain a practice, as indicated by a number of our respondents. The question that needs to be raised when adopting an AMM is, if a practice is abandoned, how would this affect the practices that are to be adopted next? And how would this affect the overall maturity? The findings from our study add more questions to the suitability of the suggestions in the AMMs.

One of the rationales for abandoning Agile practices was the influence of a person. For respondent R14, *on-site customer* was introduced by the IT manager, the person's skills and abilities were so crucial that upon his departure from the organization, the practice had to cease. Meanwhile, respondent R33 the influence of one very opinionated individual convinced the rest

Table 6. Measures of success

Category	Measure	No. of respondents
Product	Product quality	12
	Customer satisfaction	9
	Number of defects/bugs	2
	Number of relevant working products/deliverables	2
	Other: Number of newly acquired users, code quality, code change quality, business value	1 each
Process	Time to deliver	16
	Cost	3
	Delivery frequency/cadence	3
	Lead time	2
	Ease to track progress (transparency)	2
	Other: Time to resolve defects, time to implement change, correct use of development process, effective use of Agile practices, number of development issues, amount of maintenance work, number of story points, number of released new features, number of met sprint goals, non ad-hoc development process, velocity	1 each
Resource	Team spirit (happiness)	6
	Budget conformance	2
	Productivity	2
	team autonomy	3
	Other: Collaboration, stress level, team engagement, ownership, mutual understanding, continuous learning, collective ownership	1 each

of the team to stop using 13 Agile practice. The case reported by R14 and R33 shows the presence of a "maverick" [31], a highly competent and influential individual that can influence the introduction and abandonment of Agile practices. The AMMs generally suggested that Agile practices are to be introduced in certain orders, and do not provide details on how these practices are to be introduced or sustained. This indicates that the AMMs have not considered the social aspects and uniqueness of different software development teams.

The results of our survey and interviews also indicate that an Agile practice could be abandoned because it needed another practice to be established beforehand or concurrently. For example, *tracking progress* was abandoned because *sprint planning* was not yet used (as reported by R14). This suggests that there might be dependencies between Agile practices, which the practitioners may yet to be aware of. In such cases, it would be preferred if practitioners can turn to the AMM. However, when we look at the examples of the AMMs in Table 1, we can see that each AMM has different suggestions as to which practices are introduced at which maturity level. For example, Patel and Ramachandran [5] suggested that *tracking progress* need to be introduced at the same time as *planning game*; such suggestion may not work in favor of R14. However, Sidky et al. [6] suggest that (collaborative) *planning game need* to be introduced before *tracking progress*, which could have provided a better guideline for R14. This indicates there could be a need for guidelines. However, instead of suggesting to gradually introduce Agile practices in fixed orders, like the AMMs, more research can be directed to evaluate which Agile practices need to be introduced first, or later, given the contexts of the software teams or organizations.

The result of our survey indicates that practitioners, both who retained and abandoned one or more Agile practice perceive their Agile practice implementation to be successful. AMMs typically suggest that Agile practices should be continuously added in a certain order to achieve successful Agile adoption [4–6]. This indicates that successful Agile adoption could still be achieved without following the suggestions from AMMs. Our follow up interviews also revealed that an Agile practice could be replaced by another practice, such as a Lean practice. This shows that introducing Agile practices may not be as straightforward as what AMMs suggest. The follow up interviews also revealed a number of factors that could contribute to success, such as, management, leadership, and team members. This indicates the AMMs lack of consideration of the different situations and contexts in different software development team. This, again, raises the question on the merits of gradually introducing Agile practices in a certain order as suggested by AMMs.

Most of our survey respondents (82%) perceived that their Agile practice adoption to be successful and very successful. However, our respondents do not measure success the same way, for example, 12 respondents use product quality as a measure of success, and six respondents measure success given the team happiness. It indicates that success is perceived differently in different contexts by different respondents. A similar result is reported by Solinski and Petersen that indicate practitioners have different priorities on the perceived benefits and limitations of Agile practices [3]. The AMMs do not consider such prioritization of benefits and limitations that practitioners may have. This further highlights the limitation of a hierarchical approach to Agile adoption like the AMMs, as previously suggested by Gren et al. [32]. More research is needed to support practitioners in deciding which Agile practices are suitable for adoption given the benefits that they prioritized.

The results of our survey suggest that retaining or abandoning Agile practices can lead to a successful Agile adoption. This shows that Agile adoption is not as straightforward and gradual as suggested by the AMMs [4–6]. Practitioners may need to abandon, or very rarely pause, the implementation of one or more Agile practices. This indicates that practitioners are constantly assessing whether Agile practices are delivering the values they expected. Sidky et al. [6] included a step to assess whether to continue or discontinue the whole Agile transformation process, but not at the practice level. Practitioners might need support to systematically evaluate their state of Agile adoption so that decisions to add, modify, discontinue, or replace a practice is based on a rigorous and traceable process.

Implications towards Agile adoption guidelines. We noticed differences between the recommendations in AMMs and the results of our survey. At the same time, our survey also indicates the need for Agile adoption guidelines. Such guidelines need to take into account that Agile practices might not be sustainable and that there might be dependencies between Agile practices, as indicated by one our respondents, that suggests certain orders or combinations of adoptions. Furthermore, the situations and operating environment of software organizations may change [33]. The guidelines need to provide an appraisal means for practitioners on the benefits and limitations of adopting Agile practices, given the changing situations.

Implication towards Agile research. The results of our survey shares similarity to those of Kurapati et al. [16]. However, we also observed some differences, particularly pertaining to the adoption rate of *planning game*. The respondents in our survey indicate that *planning game* is a commonly used practice (47 out of 51 respondents), but Kurapati et al. reported the opposite. We observed that Kurapati et al. defined the practices slightly different. Their definition of *planning game* includes the presence an *on-site customer*. In our survey, we separated *planning game* from *on-site customer*. To be able to synthesize existing evidence regarding Agile practice adoption, there is a need for commonly agreed and consistent definitions of Agile practices.

The respondents in our survey indicate that *TDD* and *pair programming* are less commonly used practices. This result corroborates with past surveys such as [16] and [17]. *TDD* and *pair programming* are also less frequently abandoned. This observation is rather interesting because a tertiary literature study in Agile shows that TDD and pair programming is highly studied [34]. There are also many reports on their benefits and limitations to name a few: [35, 36]. This raised the question of whether knowing better the benefits and limitations of different Agile practices can help practitioners to make better decisions on whether to introduce a practice. Therefore once the decision is made to adopt such practices it is based on an informed decision. Thus the practices are less likely to be abandoned.

6. Conclusion

We conducted a survey on Agile practices with a particular focus on when adopted practices were abandoned. We received 51 valid answers, 40 provided detailed start and end period for the practices. We also conducted 11 follow up interviews with the survey respondents. In the following, we revisit our research questions by summarizing answers:

RQ1. What is the rate of adoption of Agile practices? The rate of adoption of each practice can be seen in Figure 4. Commonly adopted practices by our respondents were *face-to-face meeting*, *tracking progress*, and *planning game*. Comparably less commonly adopted practices by our respondents were *TDD* and *pair programming.*

RQ2. Which Agile practices have been abandoned? All 17 Agile practices included in this survey have been abandoned at some point (see Table 4). Consistent with the answer to the previous research question, the more commonly used practices, particularly *tracking progress* and *planning game*, also had high abandonment ratio. The rationales for abandoning Agile practices include lack of perceived values, the influence of a person, and team member discomfort. Agile practices were used between 6–60 months until they were abandoned. Most of our respondents abandoned practices within the first half year of the introduction. Agile practices are less likely to be abandoned by our survey respondents after three years (36 months) of use.

RQ3. What is the perceived success rate of Agile practices implementation? The adoption of Agile practices was perceived as being successful or very successful. Only one respondent perceived the Agile adoption as unsuccessful and none as very unsuccessful. The respondents used a large variety of measures of success. The following measures were used by the majority of respondents: product quality, customer satisfaction, and time to deliver. Furthermore, our survey indicates no differences in the perceptions of success between respondents who abandoned practices and those who retained them. This result indicates that some teams or organization needed to abandon some practices to achieve or maintain an overall successful adoption of Agile methodologies.

Future work. For future work, we suggest the following avenues of research: (1) examine how different Agile practices contribute to maturity (2) better understand the impact of gradually adding, or abandoning Agile practices, and (3) developing a common definition of Agile practices to ease aggregation of evidence.

Acknowledgement

This work had been supported by ELLIIT, a Strategic Research Area within IT and Mobile Communications, funded by the Swedish Government.

References

[1] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile software development methods – review and analysis," VTT Publications, Tech. Rep. 478, 2002.

[2] L. Williams, "Agile software development methodologies and practices," in *Advances in Computers*, Advances in Computers, M.V. Zelkowitz, Ed. Elsevier, 2010, Vol. 80, pp. 1–44.

[3] A. Solinski and K. Petersen, "Prioritizing Agile benefits and limitations in relation to practice usage," *Software Quality Journal*, Vol. 24, No. 2, 2016, pp. 447–482.

[4] J. Nawrocki, B. Walter, and A. Wojciechowski, "Toward maturity model for extreme programming," in *Euromicro Conference, 2001. Proceedings. 27th*, 2001, pp. 233–239.

[5] C. Patel and M. Ramachandran, "Agile maturity model (AMM): A software process improvement framework for Agile software development practices," *International Journal of Software Engineering, IJSE*, Vol. 2, No. 1, 2009, pp. 3–28.

[6] A. Sidky, J. Arthur, and S. Bohner, "A disciplined approach to adopting Agile practices: The

Agile adoption framework," *Innovations in Systems and Software Engineering*, Vol. 3, No. 3, 2007, pp. 203–216.

[7] T. Schweigert, D. Vohwinkel, M. Korsaa, R. Nevalainen, and M. Biro, "Agile maturity model: A synopsis as a first step to synthesis," in *Systems, Software and Services Process Improvement*, Communications in Computer and Information Science, F. McCaffery, R.V. O'Connor, and R. Messnarz, Eds., 2013, Vol. 364, pp. 214–227.

[8] M. Leppänen, "A comparative analysis of Agile maturity models," in *Information Systems Development*, R. Pooley, J. Coady, C. Schneider, H. Linger, C. Barry, and M. Lang, Eds., 2013, pp. 329–343.

[9] T. Schweigert, D. Vohwinkel, M. Korsaa, R. Nevalainen, and M. Biro, "Agile maturity model: Analysing Agile maturity characteristics from the spice perspective," *Journal of Software: Evolution and Process*, Vol. 26, No. 5, 2014, pp. 513–520.

[10] O. Ozcan-Top and O. Demirörs, "Assessment of Agile maturity models: A multiple case study," in *Software Process Improvement and Capability Determination*, Communications in Computer and Information Science, T. Woronowicz, T. Rout, R. O'Connor, and A. Dorling, Eds., 2013, Vol. 349, pp. 130–141.

[11] Version One, *8th Annual State of Agile^{TM} Report*, 2013. [Online]. https://www.versionone.com/pdf/2013-state-of-agile-survey.pdf (Accessed May 2018).

[12] Version One, *9th Annual State of Agile^{TM} Report*, 2014. [Online]. https://explore.versionone.com/state-of-agile/9th-annual-state-of-agile-report-2 (Accessed May 2018).

[13] Version One, *10th Annual State of Agile^{TM} Report*, 2015. [Online]. https://explore.versionone.com/state-of-agile/versionone-10th-annual-state-of-agile-report-2 (Accessed May 2018).

[14] Version One, *11th Annual State of Agile^{TM} Report*, 2016. [Online]. https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2 (Accessed June 2017).

[15] Version One, *12th Annual State of Agile^{TM} Report*, 2017. [Online]. https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report (Accessed May 2018).

[16] N. Kurapati, V.S.C. Manyam, and K. Petersen, *Agile Software Development Practice Adoption Survey*. Berlin, Heidelberg: Springer, 2012, pp. 16–30.

[17] B. Murphy, C. Bird, T. Zimmermann, L. Williams, N. Nagappan, and A. Begel, "Have Agile techniques been the silver bullet for software development at Microsoft?" in *Proceedings of the 7th International Symposium on Empirical Software Engineering and Measurement (ESEM 2013)*, 2013, pp. 75–84.

[18] M. Kropp, A. Meier, and R. Biddle, "Agile practices, collaboration and experience," in *Product-Focused Software Process Improvement. PROFES*, Lecture Notes in Computer Science, P. Abrahamsson, A. Jedlitschka, A. Nguyen Duc, M. Felderer, S. Amasaki, and T. Mikkonen, Eds. Springer, 2016, pp. 416–431.

[19] P. Ralph and P. Shportun, "Scrum abandonment in distributed teams: A revelatory case," in *The Pacific Asia Conference on Information Systems (PACIS)*, 2013, p. 42.

[20] J. Linåker, S.M. Sulaman, R. Maiani de Mello, and M. Höst, "Guidelines for conducting surveys in software engineering," Lund University, Tech. Rep., 2015.

[21] C. Robson, *Real world research*, 2nd ed. West Sussex, UK: John Wiley & Sons, 2011.

[22] P. Rodríguez, J. Markkula, M. Oivo, and K. Turula, "Survey on Agile and lean usage in Finnish software industry," in *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 2012, pp. 139–148.

[23] K. Petersen, "Is lean Agile and Agile lean?" *Modern Software Engineering Concepts and Practices: Advanced Approaches, IGI Global*, 2011, pp. 19–46.

[24] M. Kuhrmann, P. Diebold, J. Münch, P. Tell, V. Garousi, M. Felderer, K. Trektere, F. McCaffery, O. Linssen, E. Hanser, and C.R. Prause, "Hybrid software and system development in practice: Waterfall, scrum, and beyond," in *Proceedings of the 2017 International Conference on Software and System Process*, ICSSP, 2017, pp. 30–39.

[25] A. Forward and T.C. Lethbridge, "A taxonomy of software types to facilitate search and evidence-based software engineering," in *Proceedings of the 2008 Conference of the Center for Advanced Studies on Collaborative Research: Meeting of Minds*, CASCON '08, 2008, pp. 14:179–14:191.

[26] J. Saldaña, *The Coding Manual for Qualitative Researchers*. SAGE Publications Limited, 2012.

[27] N. Fenton and J. Bieman, *Software Metrics: A Rigorous and Practical Approach*, 3rd ed. Boca Raton, FL, USA: CRC Press, Inc., 2014.

[28] K. Petersen and C. Gencel, "Worldviews, research methods, and their relationship to validity in empirical software engineering research,"

in *Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement*, 2013, pp. 81–89.

[29] S. Stavru, "A critical examination of recent industrial surveys on Agile method usage," *Journal of Systems and Software*, Vol. 94, 2014, pp. 87–97.

[30] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R.C. Martin, S. Mallor, K. Shwaber, and J. Sutherland, "The Agile Manifesto," The Agile Alliance, Tech. Rep., 2001. [Online]. http://agilemanifesto.org/

[31] H. Sharp and H. Robinson, "Some social factors of software engineering: The Maverick, community and technical practices," in *Proceedings of the Workshop on Human and Social Factors of Software Engineering*, HSSE '05, New York, NY, USA, 2005, pp. 1–6.

[32] L. Gren, R. Torkar, and R. Feldt, "The prospects of a quantitative measurement of Agility: A validation study on an Agile maturity model," *Journal of Systems and Software*, Vol. 107, 2015, pp. 38–49.

[33] I. Nurdiani, S.A. Fricker, and J. Börstler, "An analysis of change scenarios of an IT organization for flexibility building," in *Proceedings of the 23rd European Conference on Information Systems (ECIS 2015)*, 2015.

[34] I. Nurdiani, J. Börstler, and S. Fricker, "The impacts of Agile and lean practices on project constraints: A tertiary study," *Journal of Systems and Software*, 2016.

[35] A. Causevic, D. Sundmark, and S. Punnekkat, "Factors limiting industrial adoption of test driven development: A systematic review," in *Proceedings of the 4th IEEE International Conference on Software Testing, Verification and Validation (ICST 2011)*, 2011, pp. 337–346.

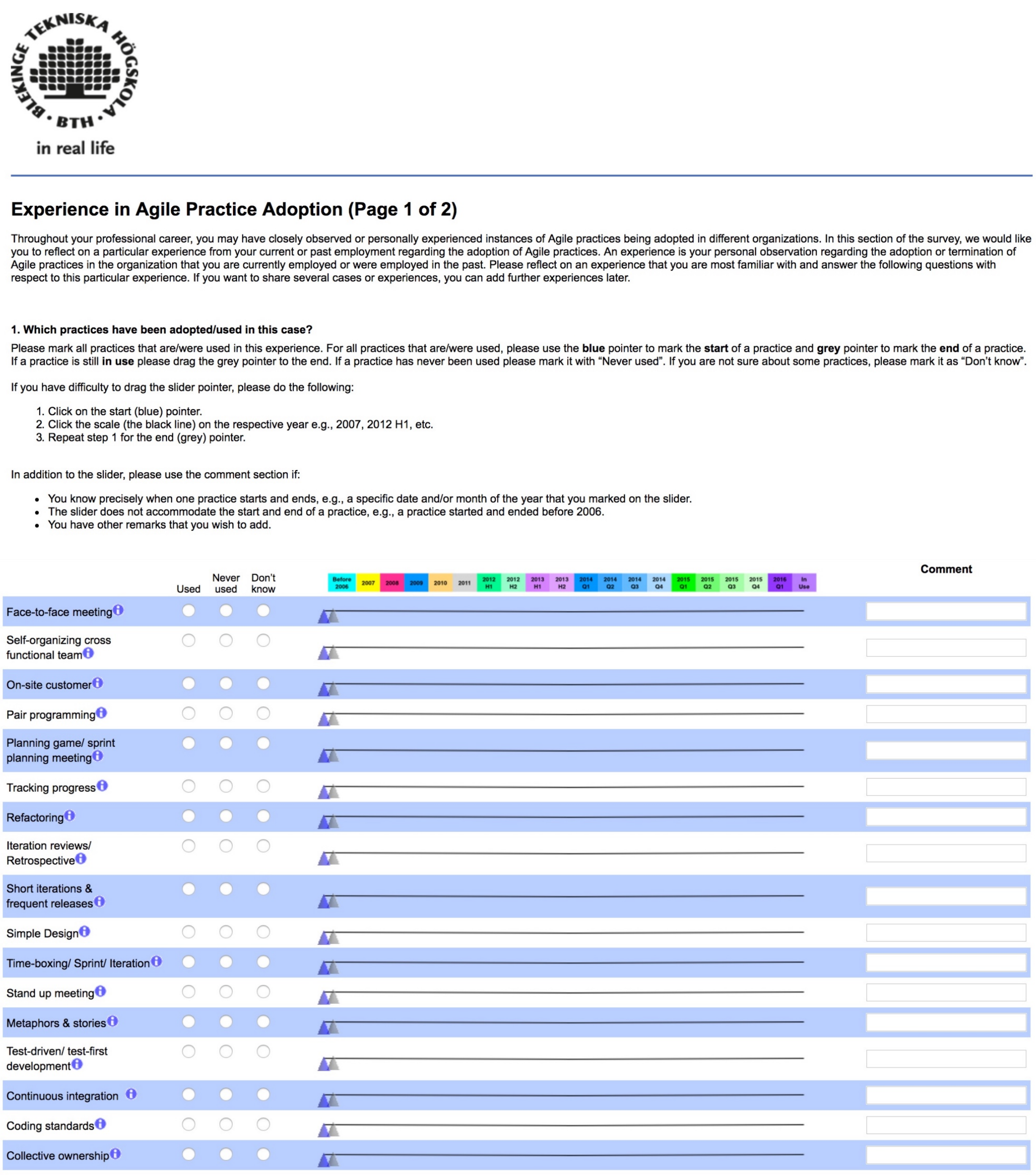
[36] T. Dyba, E. Arisholm, D. Sjoberg, J. Hannay, and F. Shull, "Are two heads better than one? On the effectiveness of pair programming," *IEEE Software*, Vol. 24, No. 6, 2007, pp. 12–15.

Appendix A. Definition of Agile practices in the survey

Agile practices definitions adapted from Solinski and Petersen [3]:

1. *Face-to-face meeting*: Team sits together, open space office facilitating interaction, video conference if the team is distributed.
2. *Self-organizing cross functional team*: Small team with no more than 10 members that consists of people with different competences (developer, tester, etc.). Team is independent, takes full responsibility of the task.
3. *On-site customer*: Continuous user involvement in the development process, customer can be consulted anytime if it is needed.
4. *Pair programming*: Two developers work together at one workstation.
5. *Planning game/sprint planning meeting*: The entire team participates in selecting the feature to be implemented in the following iteration.
6. *Tracking progress*: Tracking of the project progress using burn down chart, burn up chart.
7. *Refactoring*: Restructuring code for better understandability and reduced complexity.
8. *Iteration reviews/retrospective*: Meeting after each iteration to review the project, discuss threats to process efficiency, modify and improve.
9. *Short iterations & frequent releases*: Frequent releases of the software, early and continuous delivery of partial but fully functional software.
10. *Simple design*: Goal to design simplest solution.
11. *Time-boxing/sprint/iteration*: Fixed start and end dates are set for iterations and projects, e.g. 30 days sprint.
12. *Stand up meeting*: Short daily meeting where the whole team communicate and reflect on the completed and ongoing work.
13. *Metaphors & stories*: A metaphor is a very high level requirement outlining the purpose of the system and characterizes what the system should be like. The metaphor is broken down into short statement of the detailed functionalities called stories. The stories are kept in a backlog.
14. *Test-driven/test-first development*: Writing automated test cases for functionalities and then implementing (coding) the tested functionalities until the tests are passed successfully.
15. *Continuous integration*: Software is built frequently, even a few times a day, accompanied with testing (unit test, regression test, etc.).
16. *Coding standards*: Coding rules that are followed by the developers to make sure that developers write code in the same way.
17. *Collective ownership*: Everybody in the team can change the code of other developers in case of maintenance, bug-fixing or other development activities.

Appendix B. Survey design



Experience in Agile Practice Adoption (Page 1 of 2)

Throughout your professional career, you may have closely observed or personally experienced instances of Agile practices being adopted in different organizations. In this section of the survey, we would like you to reflect on a particular experience from your current or past employment regarding the adoption of Agile practices. An experience is your personal observation regarding the adoption or termination of Agile practices in the organization that you are currently employed or were employed in the past. Please reflect on an experience that you are most familiar with and answer the following questions with respect to this particular experience. If you want to share several cases or experiences, you can add further experiences later.

1. Which practices have been adopted/used in this case?

Please mark all practices that are/were used in this experience. For all practices that are/were used, please use the **blue** pointer to mark the **start** of a practice and **grey** pointer to mark the **end** of a practice. If a practice is still **in use** please drag the grey pointer to the end. If a practice has never been used please mark it with "Never used". If you are not sure about some practices, please mark it as "Don't know".

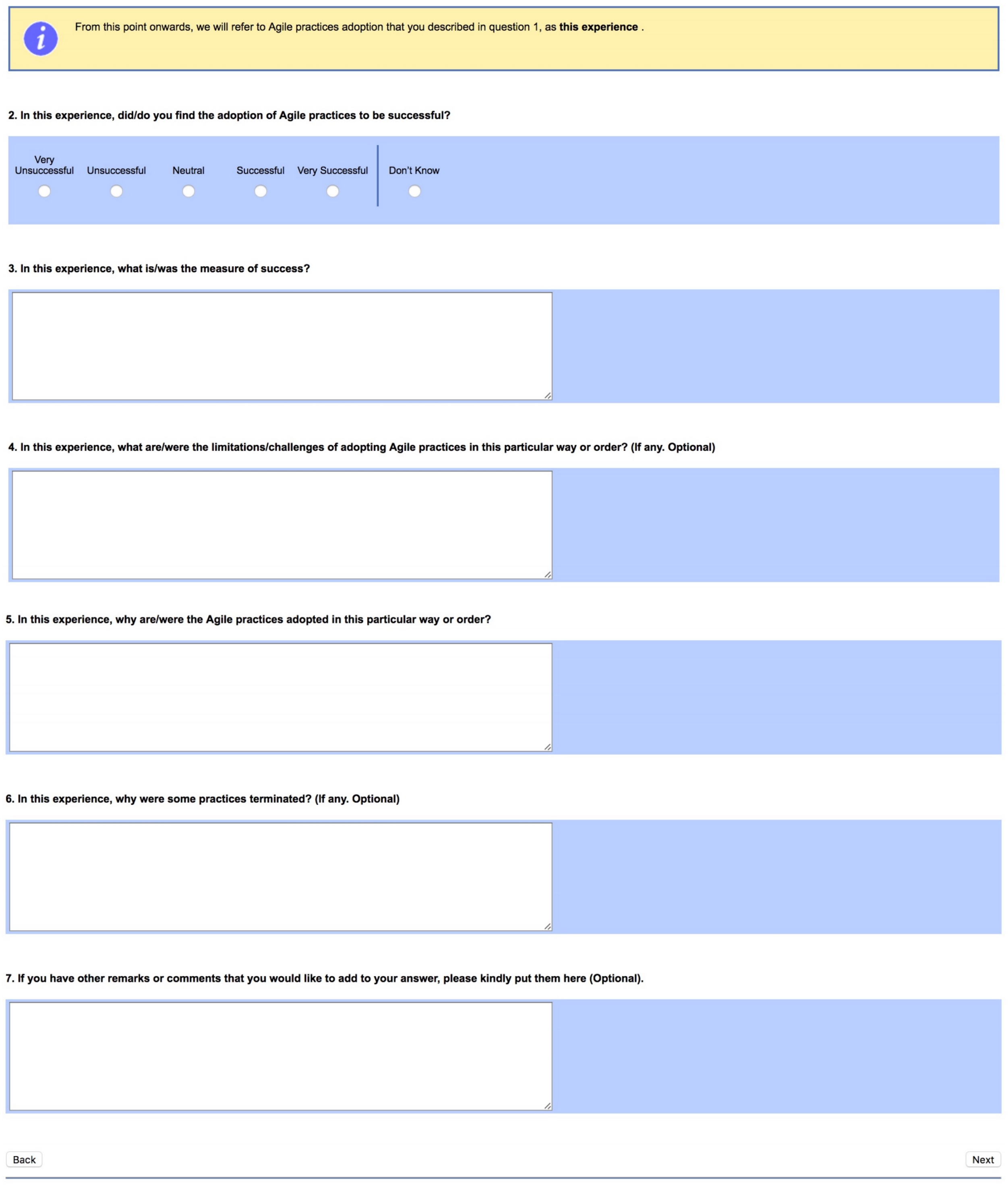
If you have difficulty to drag the slider pointer, please do the following:

1. Click on the start (blue) pointer.
2. Click the scale (the black line) on the respective year e.g., 2007, 2012 H1, etc.
3. Repeat step 1 for the end (grey) pointer.

In addition to the slider, please use the comment section if:

- You know precisely when one practice starts and ends, e.g., a specific date and/or month of the year that you marked on the slider.
- The slider does not accommodate the start and end of a practice, e.g., a practice started and ended before 2006.
- You have other remarks that you wish to add.

	Used	Never used	Don't know	Before 2006	2007	2008	2009	2010	2011	2012 H1	2012 H2	2013 H1	2013 H2	2014 Q1	2014 Q2	2014 Q3	2014 Q4	2015 Q1	2015 Q2	2015 Q3	2015 Q4	2016 Q1	In Use	Comment
Face-to-face meeting	○	○	○																					
Self-organizing cross functional team	○	○	○																					
On-site customer	○	○	○																					
Pair programming	○	○	○																					
Planning game/ sprint planning meeting	○	○	○																					
Tracking progress	○	○	○																					
Refactoring	○	○	○																					
Iteration reviews/ Retrospective	○	○	○																					
Short iterations & frequent releases	○	○	○																					
Simple Design	○	○	○																					
Time-boxing/ Sprint/ Iteration	○	○	○																					
Stand up meeting	○	○	○																					
Metaphors & stories	○	○	○																					
Test-driven/ test-first development	○	○	○																					
Continuous integration	○	○	○																					
Coding standards	○	○	○																					
Collective ownership	○	○	○																					



From this point onwards, we will refer to Agile practices adoption that you described in question 1, as **this experience** .

2. In this experience, did/do you find the adoption of Agile practices to be successful?

Very Unsuccessful	Unsuccessful	Neutral	Successful	Very Successful	Don't Know
○	○	○	○	○	○

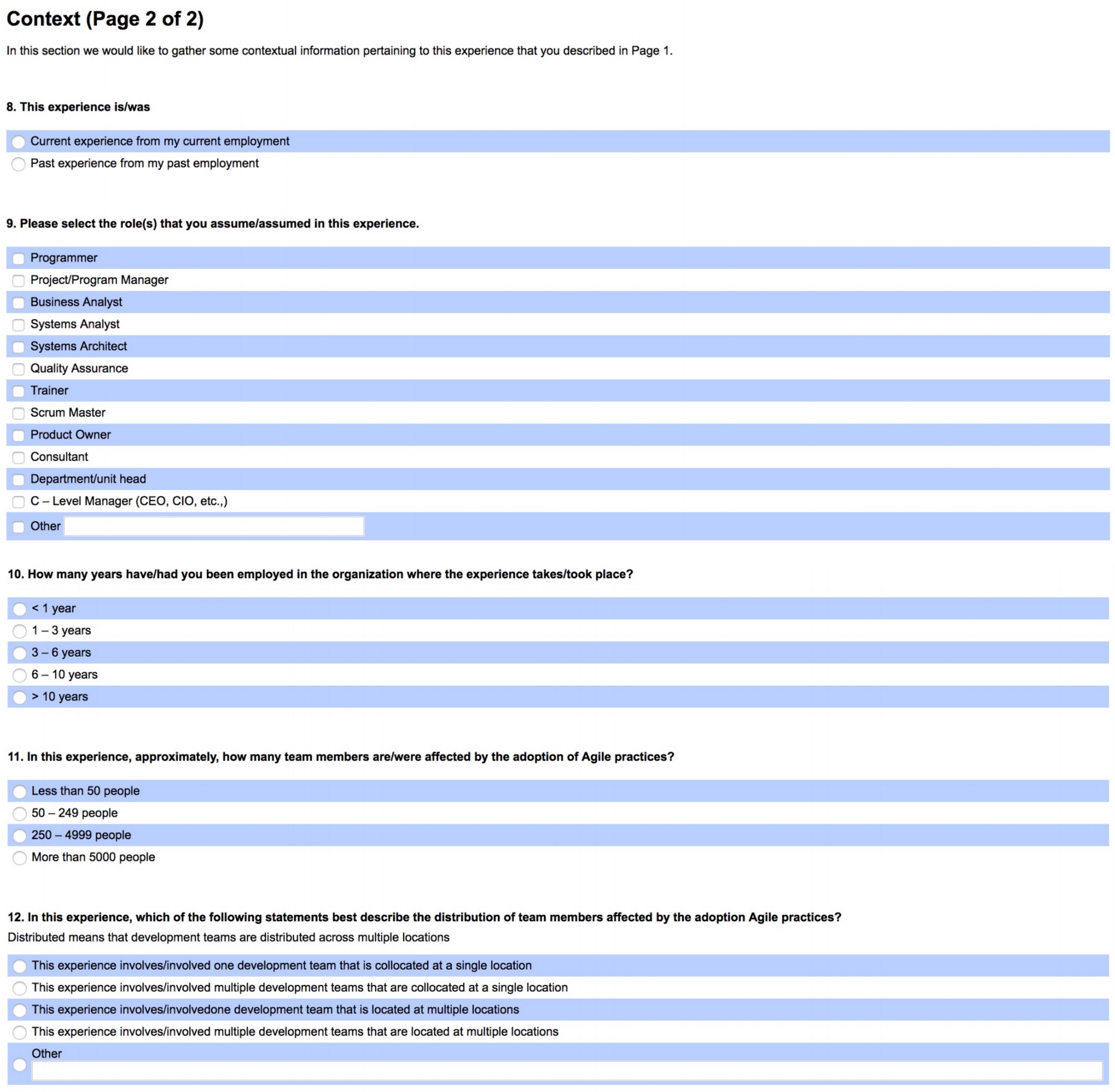
3. In this experience, what is/was the measure of success?

4. In this experience, what are/were the limitations/challenges of adopting Agile practices in this particular way or order? (If any. Optional)

5. In this experience, why are/were the Agile practices adopted in this particular way or order?

6. In this experience, why were some practices terminated? (If any. Optional)

7. If you have other remarks or comments that you would like to add to your answer, please kindly put them here (Optional).



Context (Page 2 of 2)

In this section we would like to gather some contextual information pertaining to this experience that you described in Page 1.

8. This experience is/was

- Current experience from my current employment
- Past experience from my past employment

9. Please select the role(s) that you assume/assumed in this experience.

- Programmer
- Project/Program Manager
- Business Analyst
- Systems Analyst
- Systems Architect
- Quality Assurance
- Trainer
- Scrum Master
- Product Owner
- Consultant
- Department/unit head
- C – Level Manager (CEO, CIO, etc.,)
- Other

10. How many years have/had you been employed in the organization where the experience takes/took place?

- < 1 year
- 1 – 3 years
- 3 – 6 years
- 6 – 10 years
- > 10 years

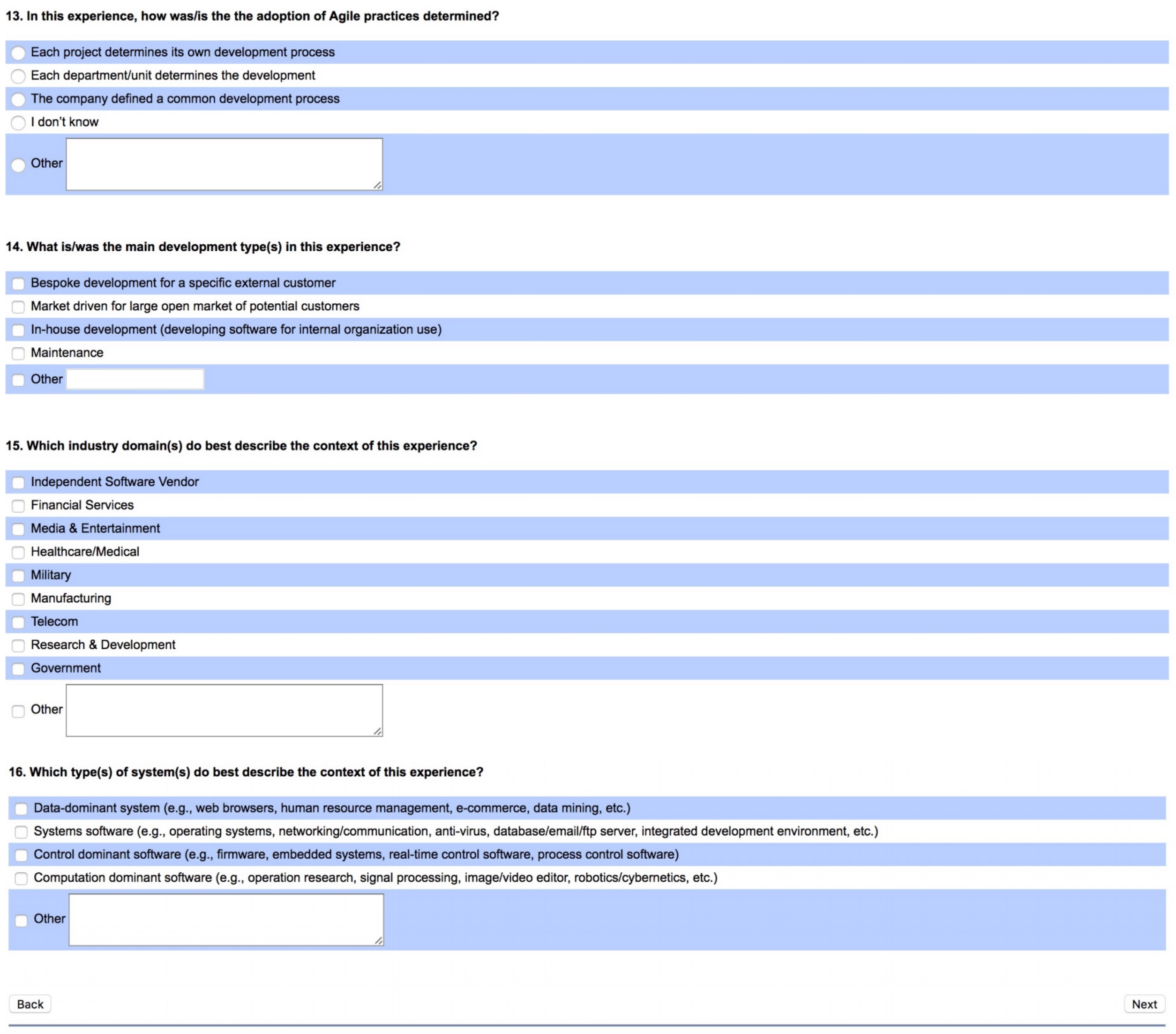
11. In this experience, approximately, how many team members are/were affected by the adoption of Agile practices?

- Less than 50 people
- 50 – 249 people
- 250 – 4999 people
- More than 5000 people

12. In this experience, which of the following statements best describe the distribution of team members affected by the adoption Agile practices?

Distributed means that development teams are distributed across multiple locations

- This experience involves/involved one development team that is collocated at a single location
- This experience involves/involved multiple development teams that are collocated at a single location
- This experience involves/involvedone development team that is located at multiple locations
- This experience involves/involved multiple development teams that are located at multiple locations
- Other



13. In this experience, how was/is the the adoption of Agile practices determined?

- Each project determines its own development process
- Each department/unit determines the development
- The company defined a common development process
- I don't know
- Other

14. What is/was the main development type(s) in this experience?

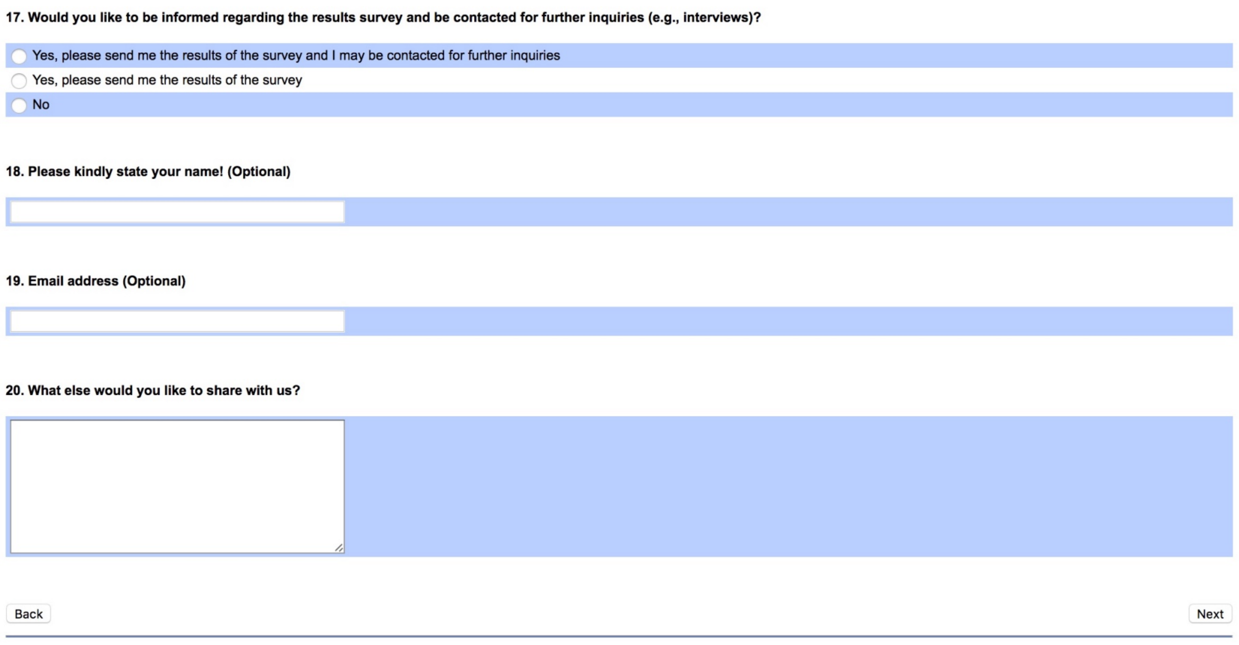
- Bespoke development for a specific external customer
- Market driven for large open market of potential customers
- In-house development (developing software for internal organization use)
- Maintenance
- Other

15. Which industry domain(s) do best describe the context of this experience?

- Independent Software Vendor
- Financial Services
- Media & Entertainment
- Healthcare/Medical
- Military
- Manufacturing
- Telecom
- Research & Development
- Government
- Other

16. Which type(s) of system(s) do best describe the context of this experience?

- Data-dominant system (e.g., web browsers, human resource management, e-commerce, data mining, etc.)
- Systems software (e.g., operating systems, networking/communication, anti-virus, database/email/ftp server, integrated development environment, etc.)
- Control dominant software (e.g., firmware, embedded systems, real-time control software, process control software)
- Computation dominant software (e.g., operation research, signal processing, image/video editor, robotics/cybernetics, etc.)
- Other


17. Would you like to be informed regarding the results survey and be contacted for further inquiries (e.g., interviews)?

- ○ Yes, please send me the results of the survey and I may be contacted for further inquiries
- ○ Yes, please send me the results of the survey
- ○ No

18. Please kindly state your name! (Optional)

19. Email address (Optional)

20. What else would you like to share with us?

Appendix C. Survey thoroughness assessment

Calculating thoroughness score. We summed up the weights for every criterion that was fulfilled by this survey (total score). Then, we divided the obtained total score by the total weight of all criteria. For more details on survey thorough assessment, see [29].

Table C.1. Survey thoroughness assessment based on [29]

Criteria	Weight	Score	Criteria	Weight	Score
Objectives	1	1	Questionnaire evaluation	3	3
Sponsorship	1	0	Questionnaire	3	3
Survey method	4	4	Media	1	1
Conceptual model	4	4	Execution time	1	1
Target population	4	4	Response burden	1	0
Sampling frame	5	5	Follow-up procedures	2	0
Sampling method	5	5	Responses	3	3
Sample size	5	5	Response rate	5	5
Data collection method	3	3	Assessment of trustworthiness	5	0
Questionnaire design	4	4	Discussions of validity threats	3	3
Provisions for securing trustworthiness	3	3			
Total weight: 66			Total score: 57		

e-Informatica Software Engineering Journal, Volume 13, Issue 1, 2019, pages: 37–62, DOI 10.5277/e-Inf190102

Do Software Firms Collaborate or Compete? A Model of Coopetition in Community-initiated OSS Projects

Anh Nguyen-Duc[*], Daniela S. Cruzes[**], Snarby Terje[***], Pekka Abrahamsson[****]

[*]*Business school, University of South Eastern Norway*
[**]*Sintef Digital*
[***]*Genus AS*
[****]*University of Jyväskylä*

Anh.Nguyen.Duc@usn.no, daniela.s.cruzes@sintef.no, terjesnarby@gmail.com, pekka.abrahamsson@jyu.fi

Abstract

Background: An increasing number of commercial firms are participating in Open Source Software (OSS) projects to reduce their development cost and increase technical innovativeness. When collaborating with other firms whose sought values are conflicts of interests, firms may behave uncooperatively leading to harmful impacts on the common goal.
Aim: This study explores how software firms both collaborate and compete in OSS projects.
Method: We adopted a mixed research method on three OSS projects.
Result: We found that commercial firms participating in community-initiated OSS projects collaborate in various ways across the organizational boundaries. While most of firms contribute little, a small number of firms that are very active and account for large proportions of contributions. We proposed a conceptual model to explain for coopetition among software firms in OSS projects. The model shows two aspects of coopetition can be managed at the same time based on firm gatekeepers.
Conclusion: Firms need to operationalize their coopetition strategies to maximize value gained from participating in OSS projects.

Keywords: COSS, coopetition, collaboration, competition, open source software, case study

1. Introduction

Increasingly, software products are no longer developed solely in-house, but in a distributed setting, where developers collaborate with "distributed collaborators" beyond their firms' boundary [1, 2]. This phenomenon includes open source software (OSS) communities, crowd-sourcing, and software ecosystems (SECO). This differs from traditional outsourcing techniques in that initiating actors do not necessarily own the software developed by contributing actors and do not hire the contributing actors. Community-initiated OSS projects are an example of the context in which actors coexist and coevolve.

From firms' perspective, it is beneficial for the development of software products whose scopes exceeds their own capabilities by leveraging external resources, exploring opportunities to enter new markets [3], performing an inside-out process [4], and employing strategic recruitments [5]. From communities' perspective, the participation in such environment probably causes firms to open up its successful products and product lines for functional extensions by external developers [1]. Instead of being exclusive and localizing product development, firms are

Submitted: 1 May 2018; Revised: 24 June 2018; Accepted: 31 July 2018; Available online: 1 October 2018

exploring different ways to invite contributions from external actors without revealing core technology, business value and customer relationships [6].

Before the full potential advantages of open sourcing are leveraged, commercial firms need to consider several concerns. At the organizational level, the firm's benefit and the community goals are not always the same [7]. Participation of commercial firms in OSS projects with their diverse motivations and business strategies might introduce variance, and sometimes conflicts in project evolution [3]. Existing research on OSS highlights the role of collaboration with extensive research on communication and coordination practices, patterns and lessons learnt from OSS communities [8–11]. However, there seems to be far less research concerns about the conflicts among firms regarding to their strategic development. Firms attempt to gain competitive advantages from their participation in OSS projects [12]. When there occur mismatches in term of interests and objectives, firms may behave uncooperatively in order to prevent others from achieving their goals [13]. The conflict occurs not only at the managerial level, such as project governance [14], but also at the operational level, such as code contribution, bug fixes, and requirement elicitation [3, 15–17].

Coopetition, as a business phenomenon, is about collaborating and handling a firm's competitive advantages when participating in OSS projects [18, 19]. In a coopetitive environment, firms cooperate with each other to reach a higher value creation compared to the value created without the interaction. The basic assumption for coopetitive relationships is that all activities should aim at the establishment of a beneficial partnership with other firms, including partners who may be considered as a kind of competitor [20]. Since coopetition applies to inter-firm relationships, OSS project offers an ideal context for understanding the phenomenon among firms that develop and utilize a common software codebase [16].

Empirical research on coopetition is scarce, especially studies in Software Engineering (SE) and at the organizational level [13]. Research in this area is probably hidden by the inconsistent treatment of the cross-disciplinary natures of cooperation and competition, and their related constructs. Our research objective is to explore how firms interact and manage the phenomenon of coopetition in OSS projects. To best of our knowledge, there exists only a few studies that examine the phenomenon of coopetition among commercial firms in OSS projects [3, 13, 15, 17]. Research questions (RQs) were derived from this research objective. Firstly, we aimed at understanding the basic foundation on firm participation in OSS projects. Based on this knowledge, we explored further theoretical elements of coopetition. We use here the word "coopetitively" as an adverb of coopetition:

- RQ1: How do commercial firms participate in community-initiated OSS projects?
- RQ2: How do commercial firms manage coopetition with other firms in such context?

Our contributions are two folds, firstly we portrayed the situations where both competition and collaboration occurs in OSS projects. Considering the body of knowledge about firm participation in OSS projects, our work confirms some patterns and also extends them by exploring the firm awareness, coopetition and their antecedent factors. Adopting a mixed-method research, we quantitatively examine organizational interaction patterns and qualitatively explore how firms perceive and employ coopetition strategies. Secondly, we theorize constructs of coopetition by proposing a Coopetition in Open Source Software (COSS) model. Previous studies that mention the term "coopetition" [3, 15], do not investigate the constructs under this phenomenon. Hence, to our best knowledge, this is among the first studies in SE investigating this concept. The proposed model reveals building blocks of coopetition in OSS firms network and its relationship to consequent factors.

The study is organized as follows: Section 2 presents a background about coopetition and firm participation in OSS projects. Section 3 describes our research methodology, Section 4 presents our findings, and Section 5 discusses the findings. Finally, Section 6 concludes the paper.

2. Background and related work

2.1. The phenomenon of coopetition

The origin of coopetition is from business research when investigating buyer and seller relationships within a business network [18, 19]. The trade-off between cooperation and competition is emphasized as a mean of creating a progress among actors involved in long-term relationships. Coopetition conceptualizes the interaction among firms in relation to their strategic development [18, 19]. Dagnino et al. defined coopetition as "a kind of inter-firm strategy which consents the competing firms involved to manage a partially convergent interest and goal structure and to create value by means of coopetitive advantage" [21]. The authors proposed two forms of coopetition, a dyadic coopetition (concerns among two-firm relationships) and a network coopetition (involving more than two firms, i.e. value chain) [21]. Bengtsson argued that a dyadic relationship is a paradox that emerges when two firms cooperate in some activities, such as in a strategic alliance, and at the same time compete with each other in other activities [19]. It means that actors within a firm need to be divided to take charge of either collaboration or competition.

Coopetition can occur in a more complex form, with a network of firms. The coopetition strategy can be applied at a micro level (among functional and divisional departments in a firm), a meso level (among firms in the same industry, between vendor and supplier) and a macro level (among cluster of firms or firms across industries) [21]. Literature also discusses some antecedent factors relating to coopetition at the micro level, such as shared vision, perceived trust and perceived benefits [22]. A study points out some possible impacts of coopetition on knowledge sharing and job/task effectiveness [22]. By selecting a highly innovative OSS project that contributes to firms' strategic values, we illustrate dependencies between competitors due to structural conditions, why and how competitors cooperate.

2.2. Collaboration in OSS projects

Collaboration is an aspect of coopetition that is much explored in OSS projects. It is common to look at OSS projects' archives to reveal communication, collaboration and coordination approaches, frequency, patterns and best practices at different level of analysis [2, 23–31]. Early research has observed an onion-like structure of contribution in OSS projects [24–27]. At the center of the onion are the core developers, who contribute most of the code and take care of the design and evolution of the project. In the next ring out are the co-developers who submit patches (e.g. bug fixes), which are reviewed and checked in by core developers [28]. Further out are the active users who do not contribute code but provide use-cases and bug-reports as well as testing new releases. The awareness of people and activities through OSS social structures enhances collaboration effectiveness and ensures that little effort is wasted in duplicate work [30]. A large amount of studies investigates the combination of social and technical aspects of OSS projects, by analyzing a social network created by contributors who work and communicate in the same set of files [32–35]. Bird et al. [34] showed that a socio-technical network of software modules and developers is able to predict software failure proneness with greater accuracy than other prediction methods. Wolf et al. [35] formed a developer-task network to explore the impact of developer communication on software build integration fail. A common assumption of these studies is that developers behave regardless of their commercial affiliations in OSS projects, indicating by unweighted analysis approaches when formulating the social networks. In case a significant number of developers from firms contributes to the project, organizational features, such as firms' strategies and governance mechanism might influence the communication structures of the OSS projects. In this work, we will use the social network analysis (SNA) to investigate interaction patterns, i.e. collaboration and competition in OSS projects. While we also form the developer-task-developer network, the difference is that the relationship is analyzed at firm level.

2.3. Collaboration in OSS projects

A theoretical model links theoretical elements in a certain semantic manner, i.e. a causal relationship, helping to design data collection and analysis. Literature reveals factors that lead to the occurrence of collaboration and competition (antecedent factors), and their impact on firms' outcomes (consequent factors). It is noted that we do not aim for model completeness, but for a foundation of further investigation. The further investigation would discover which factors valid in the context of software industry, particularly OSS projects.

As seen in Figure 1, coopetition is the studied construct, and it is linked to its antecedent factors, i.e. structural condition, strategic vision, trust and perceived benefits [22, 36–41].

Strategic vision: sharing strategic vision is essential for cooperation at team level [22] [35], as the vision reflects important agreements of beliefs and assumptions that consequently bring internal stability to the cooperative attitude [36]. At the strategic level, vision typically is about the firm's value and business development. Shared vision draws a roadmap for the organization or firm, setting the priorities for their team planning and implying its critical determinant role in lessening malign competition [22]. The vision can be shared via meetings or workshop with high-level managers.

Trust: is considered as a relationship of reliance among members of a team or an organization. Trust is defined as "the willingness of a party to be vulnerable to the actions of another party based on the expectation that the other will perform a particular action important to the trustor, irrespective of the ability to monitor or control that other party?" [42]. The importance of trust in the success of interpersonal relationships is reported previously in OSS projects [37, 38]. Moreover, trust is the key of transforming OSS as a community of individual developers, to OSS as a community of firms [39]. The cooperation that captures the level of coordinated actions between team members in their efforts to achieve mutual goals cannot be realized without trust among the members.

Perceived benefit: on one hand, perceived benefits are associated with a cooperative attitude, involving compatible interests as common benefits can motivate collaboration, leverage team or person's capabilities for obtaining such benefit [40]. In OSS projects, perceived benefits of participating in the communities are reduced development cost, community knowledge, and reduced maintenance cost. On the other hand, perceived benefit is also associated with a competitive attitude. Individuals are likely to pursue their own objective at the expense over all team's goal [41]. This could be applicable for organization in an ecosystems or supply chains. The more benefit a firm perceive for obtaining a conflicting artifact or resource, the more they likely to compete over the resource [22].

In our theoretical framework, coopetition is also associated to its consequent factors, i.e. knowledge sharing and task effectiveness [22, 43].

Knowledge sharing at organizational levels is seen as sharing of organizational experience and knowledge, i.e. technical know-how, domain expertise, work practice, etc. with other collaborators, and hence increasing the overall knowledge in the joint project [22]. As knowledge is a critical source of competitiveness, managing knowledge sharing among members of an organization plays a prominent role in sustainable competitive advantage [43].

Task effectiveness in team collaboration represents individuals' perceived capacity of conducting collaborative tasks, whereas knowledge sharing enhances the ability of collaborator's knowledge exchange.

3. Research approach

3.1. Study design

We conducted this work by using a two-phase multiple-case study design [44]. The phases in the research occur due to the discrete continuation of our internal research project. Compared

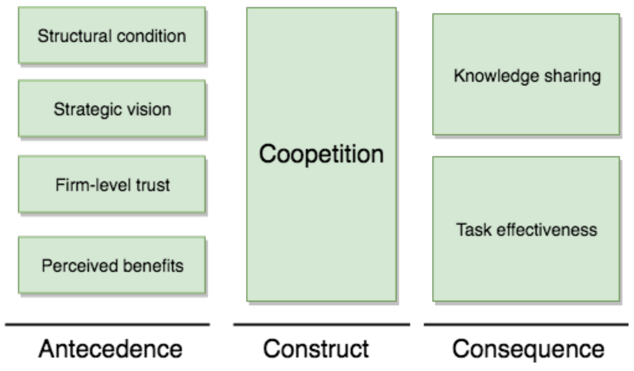


Figure 1. A theoretical framework of coopetition (adapted from [22])

to descriptive and confirmative case studies, exploratory case studies are suitable for the first phase research as we would like to discover the phenomenon of coopetition, whether it exists, in which form and its relationship to its context setting. This phase was done as a part of a master thesis. In the second phase, we conducted a descriptive study on describing collaboration, competition in the selected cases. In the third phase, we found another case study to confirm the qualitative findings. This step was conducted to validate what we observed in the first two cases. We followed the guideline by Runeson and Höst [45] to execute case study, including case selection, data collection and analysis.

Case selection is not straightforward. There are abundant OSS projects available; many of them are abandoned or individual efforts. A brainstorm session was conducted among the paper's authors to decide case selection criteria as below:

– Commercial participation: the OSS project should have multiple commercial firms participating in the development. In addition, there must be an adequate way to identify them.
– Successful and on-going: the OSS project must be successful and on-going. This implies that the project attracts developers and the development of the software is progressing.
– Active projects with many activities: the OSS project must have a high level of communication and code commits in the project, showing by rich data archive.

By reviewing literature on OSS projects in SE, we learnt several OSS projects that were commonly investigated in SE research, such as Apache, Mozilla, Eclipse and Linux [46]. The selected cases should not only satisfy the selection criteria, but also novel in SE research. We were suggested to Wireshark by a colleague who participated in the project. Many reasons contributed to this choice. Firstly, the contributor list and community activity revealed high participation and involvement of commercial companies. Wireshark is a typical instance of a OSS project. The project uses software informalisms for development collaboration, the developers are a mix of firm-paid developers and volunteers, and the software is licensed under the GNU General Public License (GNU GPL). Wireshark is also a very successful on-going OSS project, with a high number of contributors and active users, consistently pushing development forward. Having selected Wireshark as the first case, we proceeded to find and select the second case for our study. To be able to do a literal replication, the second case should have similar properties as the first case. After a long period of searching, we ended up with three promising cases that matched the specifications: Horde, Samba and Wine. From the comparison it was evident that Samba was very similar to Wireshark, i.e. both projects were licensed under GNU GPL, both projects had many firms participating, and they both had a yearly conference where developers cane together to discuss further development and socialize. We planned to have the third case to validate the qualitative findings from Wireshark and Samba. Among several OSS projects we

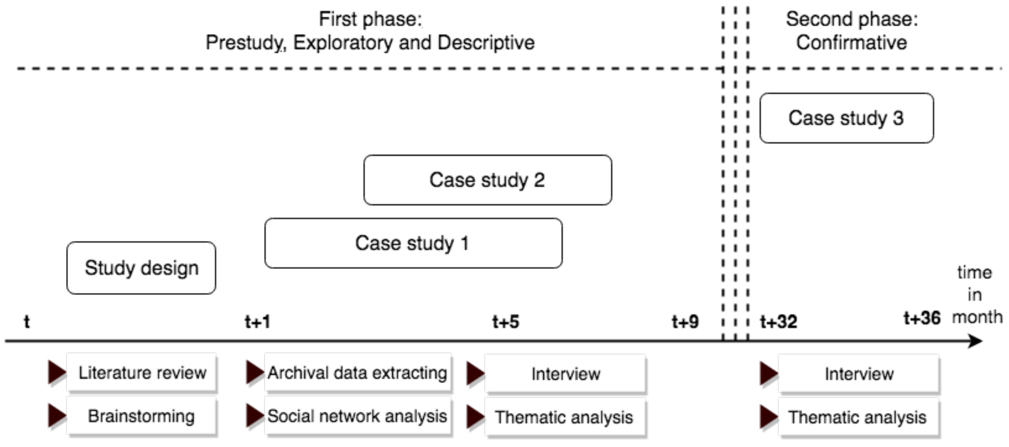


Figure 2. Overview of the research process

attempted to contact, Bootstrap developers were the one agreed to participate in the study.

The research process is described in Figure 2. At the pre-study phase, literature review and brainstorming with experts were done to come up with research objective and study design. At the exploratory and descriptive phase, the first two cases were investigated for understanding how commercial firms participated in OSS projects, if the phenomenon coopetition exists and in which form. As the explorative nature of this phase, a wide range of topics was discovered, such as collaboration patterns, firm awareness, competition, code practices, etc. The data were extracted from project archive, i.e mailing lists, bug tracking system and code repository. In this phase, we also collected qualitative data, i.e. interviewing relevant stakeholders to explore in-depth phenomenon observed from the quantitative data. At the confirmative phase, we conducted some interviews to confirm and to validate the observation from the first two cases.

3.2. Case description

Wireshark[1] is an OSS toolkit developed by a community of networking experts around the world under the GNU General Public License. The project is officially operated under the Wireshark name since May 2006. Out of the 802 developers listed in Wireshark contributor list, 342 were classified as firm-paid developers (43%). The remaining 460 developers (57%) were classified as volunteering developers. The firm-paid contributions come from 228 firms.

Samba[2] is an OSS suite that provides file, print and authentication services to all clients using the SMB/CIFS protocol. Samba is licensed under the GNU General Public License, and the Samba project is a member of the Software Freedom Conservancy. In Samba, 316 developers were evaluated, where 182 (57%) of them were classified as firm-paid developers. The contributions come from 45 firms. Communication and collaboration between developers in the Wireshark and Samba community mainly occur in two places; the developer mailing list and the bug tracking system.

Later, a third OSS project was selected as a more recent project to provide complementary qualitative data. Bootstrap[3] is a frontend Javascript-based framework for developing responsive, mobile first projects on the web. The project was released as an OSS project since 2011 under MIT license. Bootstrap were contributed by large firms, such as Twitter and GitHub. At the time the research was conducted, Bootstrap has been the most-starred project on GitHub, with over 90.000 stars and more than 38.000 forks. The communication in Bootstrap was done via

[1]https://www.wireshark.org
[2]https://www.samba.org
[3]http://getbootstrap.com

many channels, i.e. StackOverflow, Slack, and GitHub tracker. Source code and issue management was done via GitHub.

3.3. Data collection

3.3.1. Quantitative data

The main source of quantitative data is from mailing lists, code and issue repositories, as they are common data sources when studying OSS [8, 15, 23, 47]. We collected three types of data, namely developer profile, firm profile and communication data. The developer profile was found from project public pages, such as project wiki and confluence page. Basic information, like developers' email addresses and timestamp of file commits were extracted from JIRA and GIT. From developers' profiles, we were also able to identify the list of firms in a OSS project. An invitation for interview was sent in a snowballing manner. After firm-paid developers accepted our invitation for interview, basic information about the firm was required by us. Besides, firm information was also collected from online sources, such as company website, and published materials. The communication data was collected from two main sources, namely issue tracking system and mailing list. These sources contained detailed information about events and activities that had occurred in the communities several years back in time. Table 1 gives an overview of when the sources were first used and how many entries they have today in Wireshark and Samba.

3.3.2. Identification of firm participation

Information whether a participant is a firm-paid or volunteer developer, is not generally available in OSS projects. Consequently, we needed to come up with a classification technique to identify firms' participation. The approach has been successfully used in a previous study [48]. The following information was evaluated in the process of classifying the developers:

- Current status in the community: active or not any more.
- Email domain: The email domain used by a developer can reveal firm association. We regard it as unlikely that a developer use a job email to participate in an OSS project if it is not related to the job as a paid developer. This measure is the most distinctive classification entity.
- Email signature: Some developers have their employment firm name as part of their email signature, which they use when posting to the mailing list or bug tracker.
- Personal homepage: Searching for a developer's name on the web can give directions to a personal homepage or blog that might reveal company association.
- Social networks: Searching for a developer's name on social networks like LinkedIn and other professional pages might reveal firm affiliation.
- Presentations and conferences: Developers that give presentations commonly include name and firm in the presentation slides, which are easy to find by a web search.

Some issues were faced when identifying contributors' affiliations. Firstly, there is a different level of contributions in OSS projects. There is often a lack of information about what is required to become a contributor. Moreover, majority of the participants in the mailing list only posted one mail, which makes it a waste of time and effort to identify these participants as the contribution towards the firm's interaction and software development is minuscule. We decided to exclude developers with less than ten entries in the mailing list or bug tracking system. Secondly, matching name, alias and email address is not always straightforward. In Wireshark, the spam protection policy hides the full email address, for instance: "From: [developer name] <name@xxxxxxxxxxxxx>". Moreover, entries in the bug tracking system have email listed, but no name. The code repository entries in Wireshark does not contain name or mail of the developer, instead a username or a nickname is used. We had to use project wiki pages and personal contacts with some core developers of the project to provide mapping of most of the usernames to the actual developers.

Table 1. Summary of quantitative data from Wireshark and Samba

Project	Data source	Date of first entry	# of entries
Wireshark	Mailing list	31.05.2006	27230
	Bug tracking system	08.04.2005	7862
	Code repository	16.09.1998	42794
Samba	Mailing list	03.01.1997	90588
	Bug tracking system	24.04.2003	9659
	Code repository	04.05.1996	84699

3.3.3. Qualitative data

Regarding to qualitative data, interviews were selected from a convenient sample consisting of the firm-paid developers from Wireshark, Samba and Bootstrap. Ten interviews were conducted as seen from Table 2. In Wireshark and Samba, we managed to have interviews from firms in a core layer and a peripheral layer (detail as shonw in Figure 6). Due to non-disclosure agreements, we did not reveal the actual identity of companies (quantitative data was publicly available, hence did not have this constraint). We used alias D1 to D10 to represent for such firms.

As we did not know much about the population, we aimed for a non-probabilistic sampling technique using a conjunction of purposive and snowball sampling. In Wireshark, we used an existing connection to one of the core contributors as a starting point, and asked for suggestion of developers that could be interesting to interview next. The core contributor pointed out relevant developers for the research topic, and assisted in contacting them by posting our interview invitation on the core contributor mailing list. In Samba, we selected relevant developers in the OSS project based on the quantitative data and sent interview invitations to these by email. In Bootstrap, we had a developer actively contributing to the project in our personal network. From him, we got two more interviews with firm-paid participants in Bootstrap.

The interview guide consisted of both closed and open questions. The closed questions were mainly used in the introduction phase of the interview to solicit background information about the respondent, firm and OSS project context. In addition, the closed questions were used to confirm or attribute statements given by other developers. The open questions were used to collect information about: (1) work process/bridge engineer role, (2) firm awareness/organizational boundary and (3) position in the community/contributions. The interview guide and interview questions is publicly available. The interviews were conducted in English, except for one in Norwegian. The duration of the interviews ranged from 45 minutes to 72 minutes. All the interviews were recorded to facilitate subsequent analysis and minimize potential data loss due to note-taking. These recordings were thereafter transcribed verbatim. Transcribing audio records resulted in 55 pages of rich text.

3.4. Data analysis

3.4.1. Social network analysis (SNA)

SNA is a common approach to investigate communication and collaboration patterns based on data from mailing lists or issue tracking systems [32–35, 49]. This has been extensively used for constructing a developer-task network and measuring different network features [32–35]. We adopted this approach in firm level, to understand the collaboration pattern among firms via communication networks. Consequently, we used the firms as nodes and the interaction between firms as edges. Interaction among firms is represented by communication via either a mailing list or comments on an issue tracking system. The SNA was done in four steps:

– Step 1: Construct discussion trees from a mailing list and an issue tracking system. A discussion tree consists of an identifier node, a source node and a set of responder nodes (which can range from none to many). The developer that initiates a discussion is regarded as the source, and the developers that

Table 2. Summary of interview profiles

Alias	Domain	Firm type	Firm size	OSSs
D1	Telecommunication	Corp.	10000+	Wireshark
D2	Wireless networking services	SME	18568	Wireshark
D3	Messaging system	SME	11 to 50	Wireshark
D4	Telecommunication	Corp.	10000+	Wireshark
D5	IT security services	Corp.	51 to 200	Samba
D6	Server and OS development	Corp.	10000+	Samba
D7	Telecommunication	Corp.	10000+	Samba
D8	Social media	Startup	43374	Bootstrap
D9	Hosting and file sharing	SME	51 to 200	Bootstrap
D10	Social media	Startup	43374	Bootstrap

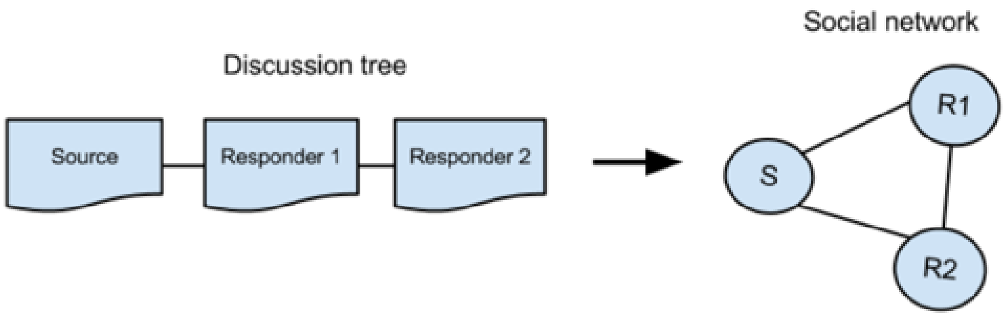


Figure 3. Constructing SNA from a discussion tree

follow-up on a discussion is regarded as responders.

– Step 2: Filter the discussion trees to remove messages with noises (irrelevant information). As shown in Figure 3, we convert a discussion tree to an undirected graph.
– Step 3: Give firm's affiliation to nodes in the graph, so that the interaction could be grouped at a firm level, rather than at individual level.
– Step 4: Build the social network by using NodeXL tool.

We were interested in the position of a firm within the context of the entire network, leading to the adoption of metrics, i.e. degree centrality, betweenness and closeness [49]:

– Degree of centrality is a measure of the number of links incident upon a firm, i.e. how many other firms that a firm is connected to.
– Betweenness centrality is a measure of the number of a shortest path between two firms that a firm lies on, quantifying the degree to which an individual in a network mediates information flow.
– Closeness centrality measures the distance from a firm to all other firms in the network. Lower values indicate that the component is farther away from all other nodes.

3.4.2. Qualitative analysis

The analysis of the qualitative data was undertaken following a guideline for thematic synthesis [50]. Thematic analysis allows main themes in the text to be systematically summarized and is also familiar by the first two authors of the paper. The process of how quantitative data from Section 3.4.1 facilitates the qualitative analysis and the use of the theoretical model to guide the analysis is shown in Figure 4. The interviews were prepared for analysis by manual transcription of the audio recordings to text documents, and the email responses were refined to transcripts of the same disposition. This resulted in 55 pages of rich text. Segments of text about firms' interaction, i.e. activities, attitudes about communication, collaboration and competition were identified and labeled. Data from the Boot-

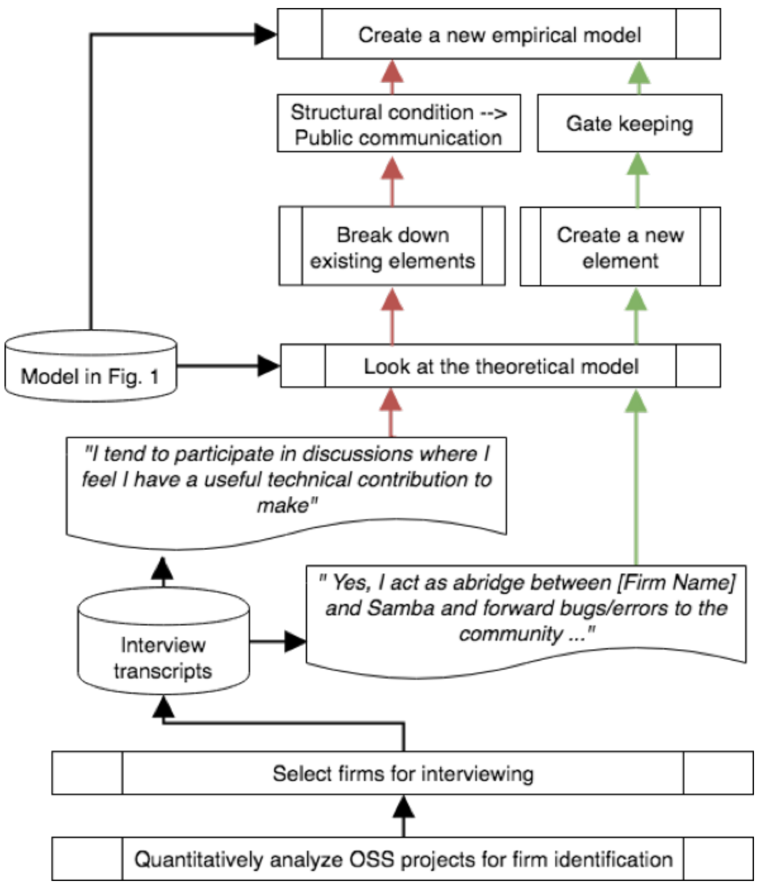

Create a new empirical model
Structural condition --> Public communication
Gate keeping
Break down existing elements
Create a new element
Model in Fig. 1
Look at the theoretical model
"I tend to participate in discussions where I feel I have a useful technical contribution to make"
Interview transcripts
" Yes, I act as abridge between [Firm Name] and Samba and forward bugs/errors to the community ..."
Select firms for interviewing
Quantitatively analyze OSS projects for firm identification

Figure 4. Steps of qualitative analysis and examples

strap case showed a level of data saturation, as there was not much new information from the case. After two rounds of reviews of the data, we ended up with 84 codes. The following step of the thematic analysis was to merge the codes and the corresponding text segments into themes. A theme in this context is essentially a code in itself, however, a theme is an increased distanciation from the text, and thus an increased level of abstraction. There are two scenarios with a theme, the first one is that identified text relates to an element in our theoretical model (as in Figure 1). The red arrow in Figure 4 describes such scenario. The second scenario is the theme could be interpreted as a new concept. The green arrow in Figure 4 describes such scenario. By grounded from existing elements and new ones, we are able to come up with an empirical model describing the concept of coopetition in three OSS projects (Section 5).

4. RQ1. How do commercial firms participate in community-initiated OSS projects?

In Section 4 we present the results of the collaboration pattern analysis. Two elements from each OSS project are presented: (1) significance of firms' contribution to OSS projects (Section 4.1), and (2) the social network structure of firms (Section 4.2).

4.1. The significance of firm's contribution

Regards to Wireshark project, from the 342 firm-paid developers, 228 unique commercial firms were identified, constituting 43% of total number of contributors. There are only 8% of the firms having three or more developers participating in the community. Firms with the largest number of participating developers are Cisco, Ericsson and Siemen. Whereas, 78% of the firms have only one developer participating. The code repository log contained 21927 entries, where 12053 of them were committed by firm-paid developers. Regards to Samba project, there are 182 firm-paid developers representing 90 different commercial firms, constituting of 58% of total number of contributors. In comparison to Wireshark project, Samba is more dominated by firms' contributions. Nine percent of total number of firms have three or more developers participating in the community, and 84% of the firms has only one developer participating. The top ten firms participating in the community with regard to number of developers is presented in Table 3.

4.2. The social network structure of firms

We illustrate the constructed SNA based on data from issue tracking systems in Wireshark, as shown in Figure 5. The node represents for a firm and the link between nodes represents for a communication link between them. The node degree was counted, including both in-degree (number of interaction received) and out-degree (number of sent interaction). By looking at the social network of Wireshark, a firm can belong to one of three contribution layers: (1) a core layer with high centrality degree, representing firms that actively communicate with others (for instance, Thales and Ericsson), (2) a peripheral layer with moderate centrality degree, representing firms with a medium number of messages to other firms (for instance, Tieto and Novell) and (3) a passive layer with low centrality degree, representing firms with small amount of message sending in and out (for instance, Broadcom and Motorola). The contribution from commercial firms in the issue tracking systems conforms to the same pattern as in the mailing list; significant, but highly diversified. In total, the issue activity by commercial firms constitute 39% in Wireshark and 66% in Samba. Figure 5 reveals that a small number of firms stay in the core layer and most of the firms locate in the passive layer. The similar network structure was observed in case of Samba project. We do not present the SNA figure for Samba due to limited space.

The collection of identified commercial firms constitutes a large fraction of the activity in the mailing list in both projects, approximately 27% in Wireshark and 47% in Samba. However, the individual firm contribution ranges from low to very high. Table 4 presents the number of messages and centrality degree of top 10 active firms in mailing list. In Wireshark project, the maximum value of centrality degree of Philips is 48, meaning that they are in contact with 48 other firms. In Samba project, the maximum value of centrality degree of Red Hat is 71, showing that they are in contact with 71 other firms. The top three firms account for 60% and 56% of the mails in Wireshark and Samba, respectively. We interviewed two firms in these lists (D1 and D5) for answering RQ2 (Section 5).

Figure 6 presents the map of our interviewed cases in the social structure of OSS projects. The selection process ensured that interviewees par-

Table 3. Summary of interview profiles

Wireshark		Samba	
Firm	# of developers	Firm	# of developers
Cisco	16	IBM	17
Ericsson	11	Red Hat	14
Siemens	8	SerNet	8
Netapp	6	SUSE	8
Citrix	5	EMC	4
Lucent	5	SGI	4
MXTelecom	5	Exanet	3
Nokia	5	HP	3
Axis	4	Cisco	3
Harman	4	Canonical	2

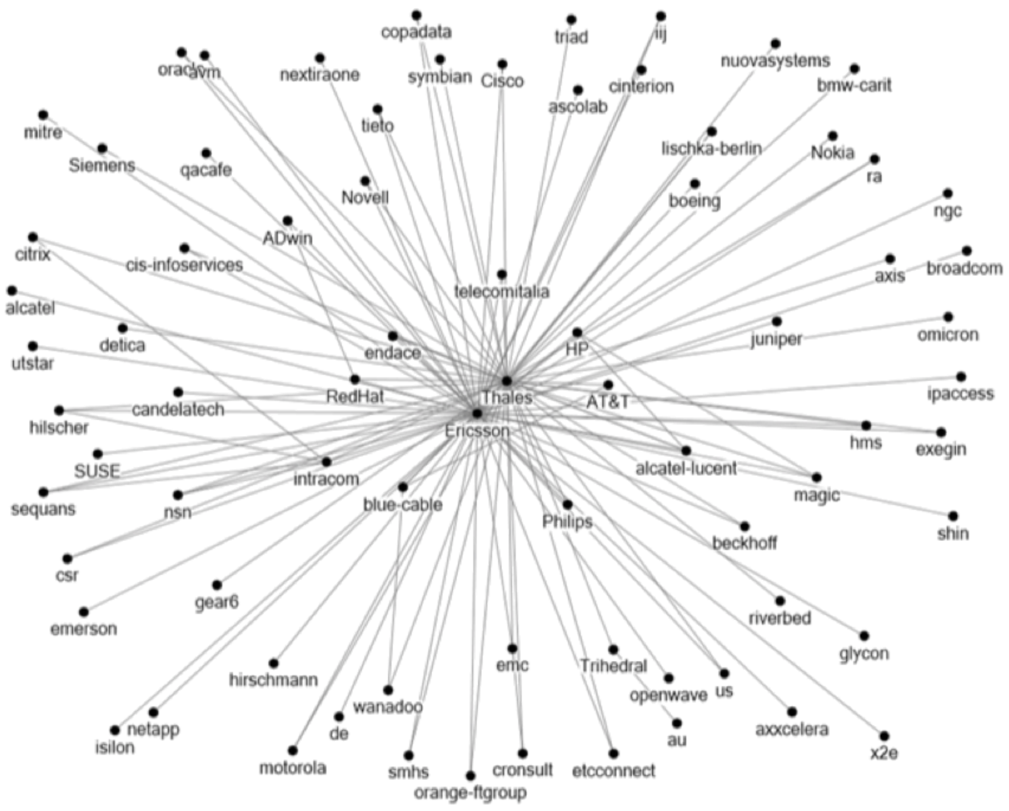


Figure 5. The social network of Wireshark via issue tracking system

ticipated in the projects for a sufficient duration. We can see that the interviewees come from different layer of the projects, hence, representing for the whole projects.

5. RQ2: How do commercial firms manage coopetition with other firms in such context?

By investigating communication patterns among firms in OSS projects and analyzing interview transcripts via the thematic analysis, we proposed a Coopetition in Open Source Software (COSS). The model is grounded from thematic concepts that extends our research presented in Section 2.3. The COSS captures the underlying phenomenon of firm participation in OSS projects from coopetition perspective. The main concepts representing the underlying phenomenon have been grouped together to form high level categories, as seen in Figure 7. The model is centralized around the concept of coopetition. Beyond the concept of coopetition in business research that consists of competition and collaboration, we identify two additional dimensions of the concept, which are gatekeeping and firm awareness. Coopetition activities are visible with the recognition of firm boundary in the projects and implemented via gatekeeping mechanisms, which are synchronizing code, strategic filtering and navigating information flow. Antecedent factors that influents coopetition concepts include structural condition, trust, perceived benefit, and strategic vision. Structural condition includes two sub concepts, public communication and direct communication. Consequent factors of coopetition include organizational learning, knowledge sharing and task effectiveness. Following sub-sections below describe the grounded evidence for each model's elements.

5.1. Public communication

The public communication channels used in our OSS projects were the mailing list and bug track-

Table 4. Summary of interview profiles

Wireshark			Samba		
Firm	Entries	Degree	Firm	Entries	Degree
Philips	1195	48	Red Hat	4480	71
Ericsson (D1)	1322	39	Sernet	3765	66
AT&T	756	34	Google	1835	57
Trihedral	222	21	IBM (D5)	1701	48
Thales	548	19	HP	1408	44
Mxtelecom	149	19	Eurocoopter	874	35
Gtech	165	13	SGI	335	29
Detica	64	10	Padl	82	29
Csr	67	10	Zylog	159	28
Sequans	31	10	Nokia	104	28

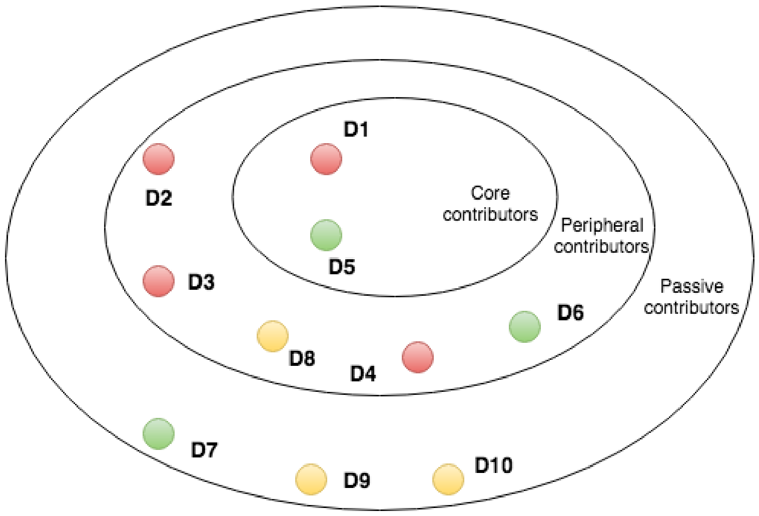


Figure 6. Social positions of interviewees in OSS projects

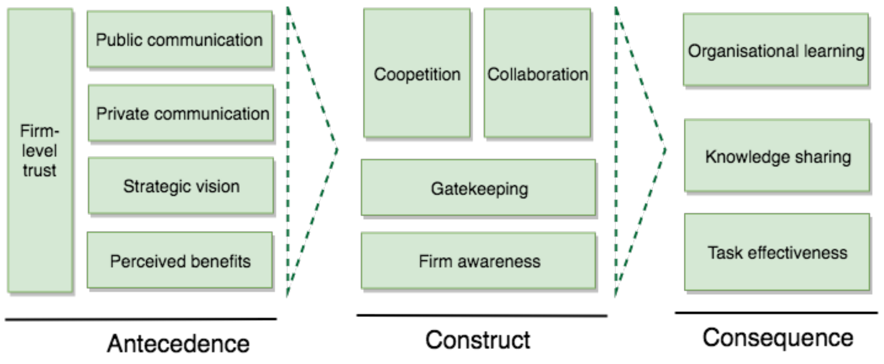


Figure 7. The model of Coopetition in Open Source Software (COSS)

ing systems. In both projects, the distribution of public communication is highly right-skewed, as shown in Figure 8. In Samba project, Sernet has contributed almost 35% of total number of message via mailing list. The top three firms account for 60% and 56% of the mails in Wireshark and Samba, respectively.

Developers mentioned several incentives for using such channels, for instance, they use the public channels for discussing, participat-

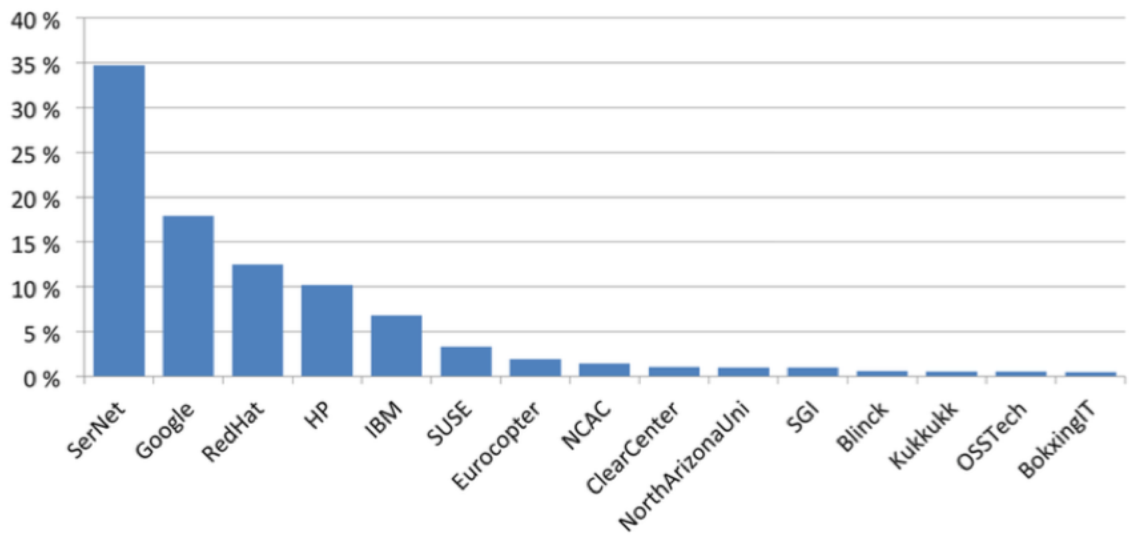


Figure 8. Distribution of number of mails per firm in Samba

ing and/or influencing the ongoing development. D4 mentioned that he publicly asked questions, discussed ideas and found collaboration via public channels: "Basically, the times when I need guidance or I have a problem, or answering other people's questions, whether it is other developers or users or whatever. Or if I have an idea about something. (...) I made a suggestion 'hey maybe we should do something to catch this problem automatically in the build-bots rather than...' Anyway, just making suggestions and putting them out basically." D6 considered mailing lists as a traceable information storage that is useful for his job: "Usually all discussions are done on the mailing list (...) this way we have a history of all discussions. I participate in discussion either to help someone with Samba or to make my point in area of my interest at the moment." Influencing project features by participation is one incentive expressed by D1: "If they are working on something that I see as usable for us internally, we find it interesting. It is smart to participate in the discussions when they are doing the development, and not come in afterwards. That is because while they are doing the changes and the development, they are more open for suggestions for changes and improvements."

Asking for guidance and support on mailing lists is common, however some developers underlined that they did not ask for solutions to their problems here. Rather, they would ask for useful advices and a push in the right direction. D3 stated that "Sometimes I have sent emails to the development list and said that I am confused by this, can someone shed some light on it." Developer D4 expressed a similar approach in: "More often I will ask people 'OK, I have this problem and I am trying to solve it. I can see two ways to solve it, does anybody have an opinion on which way is the better way?'" By this way, technical issues within a firm can be discussed and supported by external people.

D2, D3 and D4 said that they asked questions about architectural decisions in the public channels. Posting features requests or interesting ideas is also common, and some of the interviewed developers find it motivating to describe their ideas and approach to the other community members. By this way the feature expectation is communicated and other developers can come with suggestions and even join the development. D5 and D6 stated: "I tend to participate in discussions where I feel I have a useful technical contribution to make." (D5) and "I participate in discussion either to help someone with Samba or to make my point in area of my interest at the moment." (D6).

5.2. Private communication

Firms use private communication for many purposes, including both cooperative and competitive manners. Developers mentioned that they had used direct and/or private communication channels for asking for help from the domain experts in the project. Communication channels used are e-mail and instant messaging, Skype and telephone. D3 said: "I have done it [contacted developers directly] some times in the past. Not just as a general I am stuck, can you help, but because it would be an area I knew the other guy was working on." The private communication is usually the result from a gradual establishment via public communication, as mentioned by D6: "Usually I tend to do R & D tasks myself. I often seek for reviews of my work. When I need the assistance, I will go directly to a developer in the community."

Comparing to public channels, D8 considered private communication as a way to establish high-quality contact points and potential collaboration for further projects. He mentioned that a fork from project mainstream should probably include best developers in the community who are not necessarily the guy in the "onion core". It is also stated that a private channel is a quick and efficient communication medium. D9 explained that he used instant messaging for contacting developers in the community when he wanted a quick feedback. Private communication seems to be in favor comparing to public communication. D9 mentioned: "We try to address as much as we can the issues that come to us (...) Normally if we get a private message about an issue, we will take it with higher priority" D5 mentioned that when discussing legal or security sensitive issues, he used a private communication channel. The nature of such issues invokes the use of private channels as posting it in the public channels may result in security breaches or similarly bad situations. Although none of the other developers said anything about the use of direct channels for such issues, we believe that it is a common procedure in most OSS projects.

5.3. Trust

Trust is one of the fundamental traits of a successful collaborative environment [29, 51–53]. Raymond stated that "open-source culture has an elaborate set of customs?[which] regulate who can modify software, the circumstances under which it can be modified, and (especially) who has the right to redistribute modified versions back to the community" [54]. In our cases, interviewee stated that the success of OSS projects is meaningful to them. For instance, with the advance of the Wireshark tool, D4 can use it to serve for his daily work. Based on trust, developers can collaborate for the sake of their OSS project. D3 said that they have contacted trustable developers directly to avoid asking silly or dumb questions in public: "I got relationships with other developers and sometimes we don't want to ask in mailing list causes it is a really stupid question and you do not want to ask the whole mailing list, so you just ask the guy you trust". When a developer needs help to design a code or fix a bug, other developers would be willing to assist. By helping one another, developers demonstrate their skills and knowledge, which develops a positive expectation of competence and reliability. Level of trust is related to the status of the developers in OSS projects, which is evident in the following section. The observation is aligned with previous research on the role of trust in successful interpersonal relationships [37, 38].

5.4. Perceived benefits

Despite the risks associated with competitors, many firms decided to be open in sharing and synchronizing their source code with OSS communities. Source code can be synchronized with upstream development in OSS projects, for instance, described by D5: "In general, our philosophy is to develop upstream first and then back-port changes that have been approved by the upstream community into our products. We stay very involved in the communities and try to keep the differences between our packaged software and upstream software to the minimum necessary." Firms perceive benefits with such involvement as avoiding maintenance and merging issues when combining public parts of private parts of source codes. D10 illustrated for this

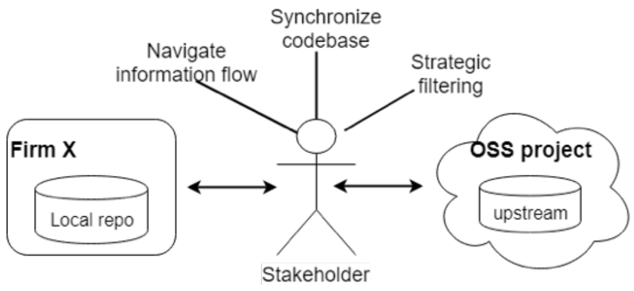


Figure 9. The role of gatekeeper in a commercial firm

idea: "…if you are to make a change in the core, and you want to keep it private, you will have to fork the project and maintain it yourself. (…) I believe, in the general case, that you gain more from contributing to the development, that retaining your code from the community". D1 mentioned that "We do not have to maintain our own code base and synchronize it. We just commit code to the source and have it there. If we had not had the commit access as easy as I do, we could have had our own version of Wireshark and the sources, but then we would have to do more work in merging our version with the new releases of Wireshark."

Firms also concern about their social positions in the projects. It is apparent that a central position in the community is closely related to being a core developer in most cases. Two benefits mentioned by the interviewees are: (1) easier code inclusion and thereby avoid the need of having a private code repository, and (2) receiving more help from other community members. D4 highlights the importance of social position in OSS community: "I think it [having a position] helps a lot. I think there is a difference if, lets say, D2 asks for help, then I will help him if I can. But if [Developer Name] from I have never really heard of, is asking for help then my level of effort is usually lower. And part of that is because I know D2 personally, and part of that is because I know that he does a tremendous amount of work. My view is that if he needs help he deserves the help. And I think it goes the other way too, if people are more likely to help me because of the contributions I have made and they know that I have been contributing for a long time. I think it helps to have some sort of status within the community."

5.5. Strategic vision

The role of strategic vision on firm participation is somewhat vague in our cases. Firm's strategy could be how a firm develops and deploy their product, i.e. how external resources are used to reduce development and maintenance cost. The vision of firm's strategy needs to be aligned at not only managerial but also operational levels. The transfer of strategic visions is not clearly evidenced in our cases. For instance, a developer D4 mentioned he spent significant office work hours as well as spare time on contributing to Wireshark. He acknowledged the benefits other developers in his firm received from his participation in the OSS project and the fact that he freely participated in Wireshark: "It is not an official part of my job, but a lot of the developers, testers and the customer support people use Wireshark extensively." However, his firm lacked formal strategies to decide how developers shall participate and develop the OSS, what code that shall be contributed back to official sources, and how to maintain the OSS knowledge base within in the firm.

5.6. Gatekeeping

The perceptions of a gatekeeper, who navigates information flows between his/her firm and external actors, were acknowledged by all interviewees. While firms might have different needs and work practices, gatekeepers are the ones stay in between the firm and the OSS project in some way, as shown in Figure 9. D1 stated that when his coworkers found issues with the third party components, they informed D1, but not project managers. D7 expressed a similar perception:

"Yes, I act as a bridge between [Company Name] and Samba and forward bugs/errors to the community." The gatekeeper is often an active actor in contributing to the community, as mentioned by D2: "Many of our core developers are working for smaller companies, and have a responsibility for the internal protocols that their company needs. (...) I think most developers work individually, and have the role of providing Wireshark functionality to the other developers in the firm."

In a cooperative manner, the gatekeeper is the hub of information and issues that can be reached by different developers across the organizations, as stated by D4: "Yes, everybody definitely knows that I am the Wireshark guy. All the developers, testers and customer support people know that they can come to me if they have Wireshark issues (...)". In firms with multiple developers active in upstream development, i.e. committing to OSS projects, there is often a recognized gatekeeper role among them. D5 mentioned: "In general when it comes to contributing patches upstream each developer in [Company Name] is independent and can directly approach the upstream project? The [Company Name] Samba package maintainer usually has a task of being the gatekeeper for those bugs that have been reported against [Company Name] products by the customers or the support teams (...)" In this case, while code is contributed independently by individuals in the firm, the bugs is managed by a gatekeeper who submits bug reports on behalf of the firm into the OSS project's bug tracking system.

In a competitive manner, gatekeepers would make sure that not all private source code be revealed to public. Firms might contribute code that relate to core components of OSS products, or utility functions. In a typical scenario, firms maintain their private repositories, where many components are parts of firms' core values. Such components should not be revealed, as mentioned by D4: "The majority of the stuff I have written for Wireshark has been pushed up? But you sort of draw a line in the stuff that is obscure enough to not push. The only people who should be looking at our proprietary protocol should be us?". Some of the code is regarded as proprietary and is retained in the firm's private code repository, due to technical specific, or legal and authorization issues D2 mentioned: "Mainly protocol dissectors for protocols used in our equipment, if the protocol is based on open protocol descriptions from 3GPP, ITU or IETF (RFC) it is considered OK to make an individual contribution to OSS (...)". Code which is not relevant, sensitive or poorly written would be filtered out by gatekeepers, as mentioned by D4: "The stuff we do not send in is stuff that is not of interest to anybody except us (...) And the other part is that I do not think the company would be thrilled by a publication of these protocols. In order to push those things to Wireshark I would need to get authorization".

5.7. Firm awareness

Several interviewees acknowledged the presence of at least another firm in the community (D1, D2, D3, D8, D9, D10). However, developers remark that it is not the knowledge of what other firms work for that is valuable, rather it is the knowledge of what business domain they are working within. D2 replied when was asked about other firm awareness: "Yes, but I do not know that much about the firms of the other developers. They typically say that they work for Firm X, and that is it. What firm they are working for is not that important to me." D3 emphasized the potential value of having the firm awareness: "(...) I know that D2 may have some role as a contact for Firm X (...) I know that D2 may be someone who is good at getting log files for specific things. In the past when I was working with voice over IP, I thought sometimes he was able to give me some log files from within his company, but I did not really think of him as the company representative. I think of him as a company person who may be able to get logs for me, like he does." In Bootstrap, developers expressed the concern on how other firms were doing related to the web technology, in order to draw lessons learnt for their product vision. D8 mentioned: "We care about if other company are using this technology in their products, so we can learn from them (...) We do not care if some guys just want to play with the technology (...)" Additionally, the interviewees were asked if they considered that their contributions could be used by other firms to gain competitive advantage. The majority dismissed this perception, for example: “As Firm X does not directly control Wireshark, I guess we have to be a bit careful when we are in contact with other developers (. . .) I believe, in the general case, that you gain more from contributing to the development, that retaining your code from the community”, stated by D2. A final remark by D5 about the competitiveness is: “Although there may be some competition between companies, as engineers we seek collaboration for mutual benefit. We already know any advancement will be used by everybody, that is not a problem, we get back as much as we give out.”

5.8. Collaboration

Although collaboration within an OSS community is typically informal and not planned, there are matters that have to be decided upon. For instance, when there is a new post in a mailing list, a developer has to decide whether to engage in the discussion with the others or not (essentially collaborating with them). The awareness of other firms in this aspect may prosper the collaboration. Firm- paid developers with similar needs and interests can collaborate and draw on each other’s abilities. Knowing that a developer works for a certain firm, and that he can provide certain code artifacts also influences the collaboration. Establishing relationships to such valuable developers through collaboration is key. There is a strong desire to return favors and honor developer’s positions by assisting them when they need help.

Many commercial firms adopt OSS, but do not participate nor contribute back to the OSS communities. Some of these firms collaborate directly with others to develop OSS-based products further, with or without participating in the OSS community. How to perform the collaboration is an aspect firms have to decide. As described above, the collaboration can take place within the OSS community using public or private communication channels, or outside the community using private channels and private code repositories.

5.9. Awareness of competition

Firms working within the same business domain are often competitors in the market, and thus it is interesting to see how influential the firm awareness is when firms come together in community based OSS projects to develop software collectively. Surprisingly, firm-paid developers said that they perceived other developers as partners and/or friends rather than competitors. D5 pointed out that he had met many developers at the OSS developer conference, and considered many of them as friends. D1 explained that he did not make any distinction between a firm-paid developer and a volunteering developer: “I think of them as developers, and not about which firms they represent.” D7 said that he would perceive others as partners. D6 mentioned: “I have always thought of others as partners. Even more – I think about them as colleagues.” D4, D8 and D9 shared similar thoughts, and dismissed the perception of other firm-paid developers as competitors: “I guess as things have evolved we do actually compete in some aspects with some of these people at this point. But that hasn’t really occurred to me much? I have noticed more people who tend to be customers of ours, rather than true competitors. We might be competitors within some areas, but I have never really thought about it I guess”, stated by D9.

The issue of competition from a firm from somewhere else in the world might not be significant for a startup and a SME who focus on having their product released as fast as possible. Without a clear vision on how their market or technical advantages are influenced by sharing and using OSS source code, the concern of competition is not much relevant. D8 also mentioned: “You think about other firms as your competitors, but I do not think that really comes in to my interactions really. They have their own users somewhere around the world (. . .) I have sometimes seen contributions from their developers, but I think that is good (. . .)”. Consequently, the coopetition concept in these OSS projects might be very much cooperation-dominant.

Another observation is that the firm’s social position is not used by any firms to dominate

OSS development. D6 mentioned: “Before working on Samba I used to think that big companies may have big influence in OSS project simply by ‘buying’ core developers. Now, that I know most of the people working on Samba, I know that this is not feasible.” Hence, having a position, or ’buying’ one, is not the way firms relate to nor influence the OSS development.

5.10. Consequent factors

Interviewees acknowledged the benefits of participating in OSS projects, including knowledge sharing, organizational learning and task effectiveness. D2 mentioned that many best practices found in reviewing code and proper comments on commits. He also appreciated the activeness level of the project with fast feedbacks. The practices are acknowledged and brought into consideration for improvement at his team. Maintaining an awareness of the other developers and what they are currently working on is also recognized and is promoted by D6 in his firms for avoiding duplicated code across the whole codebase. Organizational learning also occurs at the project level. When a firm observes the participation and interaction of core firms in the OSS projects, they can infer strategic focus areas from, i.e. feature requests and application cases.

In our cases, in-house product development depends on the OSS projects by (1) using tools as outcomes of the projects or (2) integrating and building their products on top OSS components. The dependence infers that a task that relates to OSS codes is collective performed and the task scope is beyond the OSS project. In a cooperative-dominated environment, the task will be done in an easier way. In a competitive-dominated environment, the awareness of competitors might be harmful for jointly completing the task. However, this is not directly evident from our cases.

6. Discussions

Table 5 summarizes our findings in the comparison with existing literature. While many findings confirm existing knowledge, they also provide some novel findings. This section will discuss our findings based on four topics: centralized communication structure in community-lead OSS projects (Section 6.1), modelling coopetition in the context of OSS projects (Section 6.2), the role of a gatekeeper in implementing coopetition strategies (Section 6.3) and firm contribution strategy in OSS projects (Section 6.4). Each section will discuss our findings with related work. The final section presents our actions to address threats to validities (Section 6.5).

6.1. Centralized communication structure in community-lead OSS projects

Commercial firms participating in community-based OSS projects collaborate in various ways across the organizational boundaries. Crowston et al. stated that communications structure of a project is an important element in understanding project’s practices [28]. In our cases, the majority of the activity in OSS projects is generated by a small subset of the firms, and that the remaining firms participate with little to none contribution. Wireshark and Samba demonstrate a communications centralization structure as in the onion-like social structure model [28]. Oezbek et al. [60] investigated eleven OSS projects and revealed that the role of a developer in the core layer might be more important than the fact that they do (commit code, fix bug, answer emails, etc) more. Our quantitative analysis of Wireshark and Samba confirmed these results by showing the dominant contributions of developers and firms in the core layer. Our qualitative data revealed possible importance of these developers in implementing firms’ strategies, i.e. collaboration or competition. Dewan et al. [57] showed that the heterogeneity, which exists between firm-paid developers and voluntary developers shapes the evolution of OSS community and its product. In our case, we showed that even firm-paid developers have significant contributions to code commits and communication, it is not significantly different between firm-paid and voluntary developers. From communication structure, this

Table 5. Summary of findings

Findings	Type	Current knowledge
OSS infrastructure as foundation for both public and private communication among firms	Confirmation	Structures as those in OSS enables the integration of external resources [55].
Firms activities are visible in OSS projects	Confirmation	Heterogeneity exists between firm-paid developers and voluntary developers [56, 57].
Some firms in the core positions, most of firms contribute little	New	Onion-like structure at developers level [27, 28].
Coopetition exists among firms	Confirmation	Strong explicit governance approaches can directly affect other firm's benefits [58].
Cooperation-dominated coopetition among firms at code and issue levels	Confirmation	Competition for the same revenue model does not necessary affect collaboration within OSS projects [15–17, 59].
Gatekeepers provide a mechanism to perform coopetition	Contradict	Developers within a firm need to be divided to take charge of either collaboration or competition [19].
Trust is the foundation of establishing communication, collaboration and also competition	Confirmation	Trust as a success factors in collaboration in OSS projects [38, 42].
Strategic vision is not significant at developers' level	New	Sharing strategic vision is also critical for collaboration at team level.
Firms gain social position in OSS projects, avoid merging and bug fixes, impact on influencing development and get supported	New	Perceived benefit is associated with both cooperative and competitive attitudes [22, 40, 41].

reveals a different finding from Dahlander's work [56].

6.2. Modelling coopetition in the context of OSS projects

Business literature mentions the difficulty of identifying coopetition in a real world context [21]. Dagnino et al. [21] highlighted that coopetition does not simply emerge from joining competition and collaboration, but they mix together to form a new kind of strategic interdependence between firms. We agree and illustrate for this view by showing that in OSS projects, commercial firms focus on activities that create a common value with an awareness of not sharing their technical and legal sensitive information. From our cases, COSS validates at the meso level of strategic collaboration, where firms within the same or similar domain collaborate. Among antecedent factors from literature, we highlight the role of a structural condition via public and private communication infrastructures. The transparent and effective communication infrastructure provides a mechanism for coopetition. Our study describes a competition-dominated type of coopetition. Even when firms are aware of their competitors, the attitude of collaboration is still overwhelming. Valenca et al. raise a question whether firms are collaborators or competitors in software ecosystems [3]. At the requirement engineering level, the authors found several significant challenges among firms within the same collaborative network [3]. OSS projects and firms might have divergent interests but firms can manage to discover areas of convergent interest and be able to adapt their organizing practices to collaborate [7]. In our case, this is clearly observable at the operational level. The finding also matches with observations by Linåker [15].

6.3. The role of a gatekeeper in implementing coopetition strategies

Bengtsson et al. argued that individuals within a firm could only act in accordance with one of the two logics of interaction at a time, i.e. either to compete or to collaborate [19]. Our observation on a gatekeeper role gives an alternative explanation on how firms manage such scenario. The firm's strategy can be flexible, for example fully open core sourcing at one time, and filtering of shared code at another time. The implementation of such strategies is done via the firm gatekeeper, who does actual technical contribution to the community. Therefore, in contrast with Bengtsson's findings, we find that it is possible to implement a firm-level dynamic interaction via individuals in software projects. The role of a gatekeeper is discussed in the context of commercial distributed software teams [61, 62]. Marczak et al. found the role of knowledge brokers who would have a significant impact on information flow in requirement-interdependent teams [62]. In a context of firm-to-firm interaction, we showed that a gatekeeper could navigate the information flow beyond firm's boundaries. Nguyen-Duc et al. showed four common tasks of a gatekeeper: task negotiation, conflict resolution, task- related information navigation and boundary object setups [61]. While the authors investigated gatekeepers in a software firm and a OSS project separately, this work focuses on boundary spanning activities between the OSS communities and software firms. By influencing the gatekeepers, managing code flows and information flows, firms can implement competing or collaborating strategies.

6.4. Firm contribution strategy in OSS projects

There exist some studies capturing the phenomenon of commercial firms contributing to OSS projects. Linåker et al. investigated contribution strategies of firms when participating in OSS projects [63]. The authors proposed the Contribution Acceptance Process (CAP) model to determine if source code or any types of contributions can be contributed or not. The CAP model bases on two dimensions: (1) the benefits company can receive and (2) the knowledge behind the contributions to acquire and control [63]. While these two dimensions are similar to our model's elements: perceived benefits (Section 5.4) and gatekeeping (Section 5.6), our model also explore other factors that impact the ways firms contribute to the OSS communities and collaborate with other firms. Munir et al. discussed how the openness of software firms might help them to gain benefits from OSS communities from four dimensions: (1) strategy, (2) triggers, (3) outcomes, and (4) level of openness. The model is similar with some elements in our COSS model, i.e. strategic vision, communication, gatekeeping and consequent factors. However, these models do not capture the competition strategy that firms might adopt in OSS projects. Unlike the previous work, our COSS model proposed a comprehensive view on factors that impact the strategy of collaboration and competition.

6.5. Threats to validity

6.5.1. Construct validity

Threats to construct validity consider the relationship between theory and observation, in case the measured variables do not provide a good measure of the actual factors [45]. In a qualitative study, construct validity can be thought of a "labeling" issue, as we might find the construct of the outcomes that we believe we are trying to capture. A main assumption in our study lies in the way we identify coopetition among commercial firms. As the coopetition concept comes from economic and business research, we did not have a direct map from how the concept operationalize in SE research. Previous studies that mention term "coopetition" [3, 15], do not provide the construct of this concept. Hence, to our best knowledge, this is the first study in SE attempt to operationalize this concept. We reduced this risk by a detail review and the identification of characteristics of coopetition, the exploration of the context where the construct is investigated. Both quantitative and qualitative data was collected in concept's elements and summarized in the end to describe the model. We also include discussion with co-authors and an expert in the entrepreneurship in validate our observation.

The phenomenon is operationalized based on public and private communication among developers participated in OSS projects. We were aware of other communication channels, such as private messaging, telephone and Skype, however, we do not have a feasible way to quantify this. We limited the investigation in public collaboration where developers responsed to the same mailing list or comment on the same issue. Regarding to the identification of firm participation, we used SNA with density metrics, such as degree centrality and closeness [49]. Other network-based measures for the same construct (e.g. transitivity, compactness, and connectedness) could be considered for enhancing the rigor of this research. We also used an unweight approach to perform SNA, which ignored the firms' characteristics, such as firm size, and business strategy towards the OSS community. This could be considered in future work, especially in firm-based OSS projects.

The risk of operationalization is reduced by using a mixed method research, including both quantitative and qualitative data. The intervie-wees were conducted with firms from different social position in OSS projects, which increase the credibility in the observation of phenomenon. The data is limited at ten interviews. However, we had reached data saturation [45] when interviewing Bootstrap case. Although, intervie-wees were selected from different types of OSS projects, different company profiles, we found that their responses were consistent, which increase our confidence in the trustworthiness of the data.

6.5.2. External validity

This threat considers the ability to generalize our findings. The goal of this study is not to achieve statistical generalization, but rather an analytical generalization. This is particularly important when studying a complex phenomenon, in our case is coopetition in OSS projects. To avoid the bias on findings from a single case, we analyzed two OSS projects. Qualitative data was further collected from the third OSS project to improve the generalization. With the in-depth investigation in both community and firms' sides of the projects, we are confident about the explanation power of the COSS model for similar contexts. Our OSS projects produce a library, a framework and an application, employing GPL and MIT licenses. Our cases represent for a community-initiated OSS projects, that are initiated and lead by the community. Further research should replicate our method on other types of OSS projects to explore other collaboration and competition scenarios. They are also popular OSS projects with years of operation, hence the products and collaboration process have been stable. The findings might not be directly applicable to emerging OSS projects, or projects initiated by firms. Research on projects with different types of OSS licenses might lead to a variety in our model.

6.5.3. Reliability

This threat concerns about the level to which the operational aspects of the study, such as data collection and analysis procedures, are repeatable with the same results. The main data collection was done as a part of a master thesis. All interviews were recorded and transcribed verbatim in order to make sure that no data reduction occurred prematurely. The transcription of the interviews was reviewed and interpreted by the other author. In case of vague statements, one author is responsible for follow-up discussions with interviewees for clarification. We used both quantitative data about communication among firms in the project and qualitative data from interviews of firms from different contribution layers. The data triangulation allows our findings represent the true situation of investigated projects. Moreover, the paper has gone through proof-read from several senior researchers in the domain. Their feedbacks help us to improve the paper significantly since the first draft.

7. Conclusions

Coopetition is an important topic in economics and business research [19, 21], but it is overlooked in other domains. In modern software industry, the popularity of developing software products beyond firm's boundary makes coopetition a relevant theme. In this paper, we used both qualitative and quantitative data to investigate coopetition in OSS projects. Firstly, we found that commercial firms participating in community-initiated OSS projects collaborate in various ways across the organizational boundaries. While most of firms contribute little, a small number of firms are very active and account for large proportions of contribution. It is also evident that firms interact across their boundaries in OSS projects. Secondly, we proposed an empirical model COSS to explain for root causes of coopetition in OSS projects. The COSS model shows that coopetition is based on the firm awareness, structural condition of the OSS projects and operated by gatekeepers. The coopetition is cooperation-dominated even among firms working in the same business domain with similar business models.

The findings have implications for research. We offer a descriptive explanation of how coopetition occurs and impacts in OSS projects. We observe that software firms emphasize the co-creation of common value and partly react to the potential competitiveness in OSS projects. The highlight of our findings is the COSS model, which argues that competition and collaboration can both be handled by gatekeepers. The role of gatekeepers in crossing organizational boundaries is still an interesting research topic. For SE with abundant research on OSS collaboration and communication, the study on inter-firm coopetition is a novel way of looking at the same data sources and infrastructures.

The study also has implications for practitioners. We offer software firms insights about different coopetition strategies observed in a community-driven OSS project. For instance, not all communication goes through the public channels in OSS projects. Legal and security sensitive issues commonly go through private or closed channels because of their natures. Furthermore, firms should consider a gatekeeper as an important role when they plan to participate and gain benefit from OSS projects.

For future work, the next step would be to validate the COSS model with a larger set of cases. Our research here only uses three community-driven OSS projects, which limits the generalization of findings. Moreover, a longitudinal observation on how coopetition evolves among firms can provides knowledge that goes beyond cross-sectional observations. Last but not least, further investigation about employing the role of gatekeepers for coopetition is needed to provide actionable guideline for successful operation of inter-firm coopetition. Future work can also investigate OSS project settings that affect firm collaboration, i.e. OSS license, and feature request mechanism. It would be interested to see how these factors could play a role in our model.

References

[1] D.G. Messerschmitt and C. Szyperski, *Software Ecosystem: Understanding an Indispensable Technology and Industry.* Cambridge, MA, USA: MIT Press, 2003.

[2] S. Jansen, A. Finkelstein, and S. Brinkkemper, "A sense of community: A research agenda for software ecosystems," in *31st International Conference on Software Engineering – Companion Volume*, 2009, pp. 187–190.

[3] G. Valença, C. Alves, V. Heimann, S. Jansen, and S. Brinkkemper, "Competition and collaboration in requirements engineering: A case study of an emerging software ecosystem," in *22nd International Requirements Engineering Conference (RE)*, 2014, pp. 384–393.

[4] J.F. Lorraine Morgann and P. Finnegan, "Exploring inner source as a form of intra-organisational open innovation," in *19th European Conference on Information Systems*, 2011.

[5] K. Manikas, "Revisiting software ecosystems research: A longitudinal literature study," *Journal of Systems and Software*, Vol. 117, 2016, pp. 84–103.

[6] H.H. Olsson and J. Bosch, "Ecosystem-driven software development: A case study on the emerging challenges in inter-organizational R&D," in *Software Business. Towards Continuous Value Delivery*, C. Lassenius and K. Smolander, Eds. Springer International Publishing, 2014, pp. 16–26.

[7] S. O'Mahony and B.A. Bechky, "Boundary organizations: Enabling collaboration among unexpected allies," *Administrative Science Quarterly*, Vol. 53, No. 3, 2008, pp. 422–459.

[8] B. Andrea and R. Cristina, "Comparing motivations of individual programmers and firms to take part in the open source movement: From community to business," *Knowledge, Technology & Policy*, Vol. 18, No. 4, 2006, pp. 40–64.

[9] A.H. Ghapanchi, C. Wohlin, and A. Aurum, "Resources contributing to gaining competitive advantage for open source software projects: An application of resource-based theory," *International Journal of Project Management*, Vol. 32, No. 1, 2014, pp. 139–152.

[10] A.H. Ghapanchi, "Rallying competencies in virtual communities: A study of core processes and user interest in open source software projects," *Information and Organization*, Vol. 23, No. 2, 2013, pp. 129–148.

[11] R. Riehle, "The single-vendor commercial open course business model," *Information Systems and e-Business Management*, Vol. 10, No. 1, 2012, pp. 5–17.

[12] A.H. Ghapanchi, C. Wohlin, and A. Aurum, "Resources contributing to gaining competitive advantage for open source software projects: An application of resource-based theory," *International Journal of Project Management*, Vol. 32, No. 1, 2014, pp. 139–152. [Online]. http://www.sciencedirect.com/science/article/pii/S0263786313000380

[13] S. Ghobadi and J. D'Ambra, "Coopetitive relationships in cross-functional software development teams: How to model and measure?" *Journal of Systems and Software*, Vol. 85, No. 5, 2012, pp. 1096–1104.

[14] K. Manikas and K.M. Hansen, "Software ecosystems? A systematic literature review," *Journal of Systems and Software*, Vol. 86, No. 5, 2013, pp. 1294–1306. [Online]. http://www.sciencedirect.com/science/article/pii/S016412121200338X

[15] L. Johan, R. Patrick, R. Björn, and P. Mäder, "How firms adapt and interact in open source ecosystems: Analyzing stakeholder influence and collaboration patterns," in *Requirements Engineering: Foundation for Software Quality*. Cham: Springer International Publishing, 2016, pp. 63–81.

[16] J. Teixeira, "Understanding collaboration in the open-source arena: The cases of WebKit and OpenStack," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, 2014, pp. 52:1–52:5.

[17] J. Teixeira, S. Mian, and U. Hytti, "Cooperation among competitors in the open-source arena: The case of OpenStack," in *International Conference on Information Systems (ICIS)*, 2016.

[18] A.M. Brandenburger and B.J. Nalebuff., *Co-opetition.* New York, USA: Doubleday, 1996.

[19] M. Bengtsson and S. Kock, "'Coopetition' in business networks? To cooperate and compete simultaneously," *Industrial Marketing Management*, Vol. 29, No. 5, 2000, pp. 411–426.

[20] M. Zineldin, "Co-opetition: The organisation of the future," *Marketing Intelligence & Planning*, Vol. 22, No. 7, 2004, pp. 780–790.

[21] G.B. Dagnino and G. Padula, "Coopetition strategy, a new kind of inter firm dynamics for value creation," in *The 2nd conference on European Academy of Management*, 2002.

[22] C.P. Lin, Y.J. Wang, Y.H. Tsai, and Y.F. Hsu, "Perceived job effectiveness in coopetition: A survey of virtual teams within business organizations," *Computers in Human Behavior*, Vol. 26, No. 6, 2010, pp. 1598–1606. [Online]. http://www.sciencedirect.com/science/article/pii/S0747563210001792

[23] W. Scacchi, "Free/Open Source software development: Recent research results and emerging opportunities," in *The 6th Joint Meeting on European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering: Companion Papers*, ESEC-FSE companion '07. New York, NY, USA: ACM, 2007, pp. 459–468.

[24] J.Y. Moon and L. Sproull, "Essence of distributed work: The case of the Linux kernel," *First Monday*, No. 11, 2000. [Online]. http://www.firstmonday.dk/ojs/index.php/fm/article/view/801/710

[25] A. Mockus, R.T. Fielding, and J.D. Herbsleb, "Two case studies of Open Source Software development: Apache and Mozilla," *ACM Transactions on Software Engineering and Methodology*, Vol. 11, No. 3, 2002, pp. 309–346.

[26] G.K. Lee and R.E. Cole, "From a firm-based to a community-based model of knowledge creation: The case of the Linux kernel development," *Organization Science*, Vol. 14, No. 6, 2003, pp. 633–649.

[27] J. Xu, Y. Gao, S. Christley, and G. Madey, "A topological analysis of the Open Souce Software development community," in *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, 2005, p. 198a.

[28] K. Crowston and J. Howison, "The social structure of Free and Open Source Software development," *First Monday*, Vol. 10, 2005.

[29] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in GitHub: Transparency and collaboration in an open software repository," in

Proceedings of the ACM Conference on Computer Supported Cooperative Work, CSCW '12. New York, NY, USA: ACM, 2012, pp. 1277–1286.

[30] C. Gutwin, R. Penner, and K. Schneider, "Group awareness in distributed software development," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, CSCW '04. New York, NY, USA: ACM, 2004, pp. 72–81.

[31] W. Scacchi, *Free/Open Source Software Development: Recent Research Results and Methods.* Elsevier, 2007, Vol. 69, pp. 243–295. [Online]. http://www.sciencedirect.com/science/article/pii/S0065245806690050

[32] R. Abreu and R. Premraj, "How developer communication frequency relates to bug introducing changes," in *Proceedings of the Joint International and Annual ERCIM Workshops on Principles of Software Evolution (IWPSE) and Software Evolution (Evol) Workshops*, IWPSE-Evol '09. New York, NY, USA: ACM, 2009, pp. 153–158.

[33] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schroter, and C. Weiss, "What makes a good bug report?" *IEEE Transactions on Software Engineering*, Vol. 36, No. 5, 2010, pp. 618–643.

[34] C. Bird, N. Nagappan, H. Gall, B. Murphy, and P. Devanbu, "Putting it all together: Using socio-technical networks to predict failures," in *20th International Symposium on Software Reliability Engineering*, 2009, pp. 109–119.

[35] T. Wolf, A. Schroter, D. Damian, and T. Nguyen, "Predicting build failures using social network analysis on developer communication," in *Proceedings of the 31st International Conference on Software Engineering*, ICSE '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1–11.

[36] C. Ferioli and P. Migliarese, "Supporting organizational relations through information technology in innovative organizational forms," *European Journal of Information Systems*, Vol. 5, No. 3, 1996, pp. 196–207.

[37] M. Antikainen, T. Aaltonen, and J. Väisänen, "The role of trust in OSS communities? Case linux kernel community," in *Open Source Development, Adoption and Innovation*, J. Feller, B. Fitzgerald, W. Scacchi, and A. Sillitti, Eds. Springer, 2007, pp. 223–228.

[38] S.Y. Ho and A. Richardson, "Trust and distrust in Open Source Software development," *Journal of Computer Information Systems*, Vol. 54, No. 1, 2013, pp. 84–93.

[39] P.J. Ågerfalk and B. Fitzgerald, "Outsourcing to an unknown workforce: Exploring opensourcing as a global sourcing strategy," *MIS Quarterly*, Vol. 32, No. 2, 2008, pp. 385–409.

[40] M. Bengtsson and S. Kock, "Cooperation and competition in relationships between competitors in business networks," *Journal of Business & Industrial Marketing*, Vol. 14, No. 3, 1999, pp. 178–194.

[41] D. Tjosvold, *Team organization: An enduring competitive advantage.* Wiley-Blackwell, 1991, ch. Forging Synergy, pp. 219–233.

[42] R.C. Mayer, J.H. Davis, and F.D. Schoorman, "An integrative model of organizational trust," *The Academy of Management Review*, Vol. 20, No. 3, 1995, pp. 709–734. [Online]. http://www.jstor.org/stable/258792

[43] M. Levy, C. Loebbecke, and P. Powell, "SMEs, co-opetition and knowledge sharing: The role of information systems," *European Journal of Information Systems*, Vol. 12, No. 1, 2003, pp. 3–17.

[44] R.K. Yin, *Case Study Research: Design and Methods (Applied Social Research Methods).* USA: SAGE Publications, Inc., 2014.

[45] M.J. Gallivan, "Striking a balance between trust and control in a virtual organization: A content analysis of open source software case studies," *Information Systems Journal*, Vol. 11, No. 4, 2001, pp. 277–304.

[46] H. Wang and C. Wang, "Open source software adoption: a status report," *IEEE Software*, Vol. 18, No. 2, 2001, pp. 90–95.

[47] M. Cataldo and J.D. Herbsleb, "Architecting in software ecosystems: Interface translucence as an enabler for scalable collaboration," in *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*, ECSA '10. New York, NY, USA: ACM, 2010, pp. 65–72.

[48] *Q&A with the founder of Wireshark and Ethereal*, 2008. [Online]. http://protocog.com/gerald_combs_interview.html

[49] L. Freeman, *The Development of Social Network Analysis.* Canada: Empirical Press, 2006.

[50] D.S. Cruzes and T. Dyba, "Recommended steps for thematic synthesis in software engineering," in *International Symposium on Empirical Software Engineering and Measurement*, 2011, pp. 275–284.

[51] K.J. Stewart and S. Gosain, "The impact of ideology on effectiveness in open source software development teams," *MIS Quarterly*, Vol. 30, No. 2, 2006, pp. 291–314.

[52] P.B. M. S. Lane, G. Vyver and S. Howard, "Inter-preventative insights into interpersonal trust and effectiveness of virtual communities of open source software (OSS) developers," *Open Source Systems: Towards Robust Practices*, 2004.

[53] P.B. de Laat, "How can contributors to open-source communities be trusted? on the assumption, inference, and substitution of trust," *Ethics and Information Technology*, Vol. 12, No. 4, 2010, pp. 327–341.

[54] E.S. Raymond, *The Cathedral and the Bazaar*. Sebastopol, CA, USA: O'Reilly & Associates, Inc., 1999.

[55] S. Grand, G. von Krogh, D. Leonard, and W. Swap, "Resource allocation beyond firm boundaries: A multi-level model for open source innovation," *Long Range Planning*, Vol. 37, No. 6, 2004, pp. 591–610. [Online]. http://www.sciencedirect.com/science/article/pii/S0024630104001177

[56] L. Dahlander and M.W. Wallin, "A man on the inside: Unlocking communities as complementary assets," *Research Policy*, Vol. 35, No. 8, 2006, pp. 1243–1259. [Online]. http://www.sciencedirect.com/science/article/pii/S0048733306001387

[57] A. Mehra, R. Dewan, and M. Freimer, "Firms as incubators of open-source software," *Information Systems Research*, Vol. 22, No. 1, 2011, pp. 22–38.

[58] M.J. Gallivan, "Striking a balance between trust and control in a virtual organization: a content analysis of open source software case studies," *Information Systems Journal*, Vol. 11, No. 4, 2001, pp. 277–304.

[59] J. Teixeira, G. Robles, and J.M. González-Barahona, "Lessons learned from applying social network analysis on an industrial Free/Libre/Open Source Software ecosystem," *Journal of Internet Services and Applications*, Vol. 6, No. 1, 2015, p. 14.

[60] C. Oezbek, L. Prechelt, and F. Thiel, "The onion has cancer: Some social network analysis visualizations of open source project communication," in *Proceedings of the 3rd International Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development*, FLOSS '10. New York, NY, USA: ACM, 2010, pp. 5–10.

[61] S. Marczak, D. Damian, U. Stege, and A. Schröter, "Information brokers in requirement-dependency social networks," in *16th IEEE International Requirements Engineering Conference*, 2008, pp. 53–62.

[62] A. Nguyen-Duc, D.S. Cruzes, and R. Conradi, "On the role of boundary spanners as team coordination mechanisms in organizationally distributed projects," in *9th International Conference on Global Software Engineering*, 2014, pp. 125–134.

[63] J. Linåker, H. Munir, K. Wnuk, and C.E. Mols, "Motivating the contributions: An open innovation perspective on what to share as open source software," *Journal of Systems and Software*, Vol. 135, 2018, pp. 17–36. [Online]. http://www.sciencedirect.com/science/article/pii/S0164121217302169

e-Informatica Software Engineering Journal, Volume 13, Issue 1, 2019, pages: 63–103, DOI 10.5277/e-Inf190103

Representation of UML Class Diagrams in OWL 2 on the Background of Domain Ontologies

Małgorzata Sadowska*, Zbigniew Huzar*
*Faculty of Computer Science and Management, Wrocław University of Science and Technology
m.sadowska@pwr.edu.pl, zbigniew.huzar@pwr.edu.pl

Abstract

Background: UML class diagrams can be automatically validated if they are compliant with a domain knowledge specified in a selected OWL 2 domain ontology. The method requires translation of the diagrams into their OWL 2 representation.
Aim: The aim of this paper is to present transformation and verification rules of UML class diagrams to their OWL 2 representation.
Method: The analysis of the results of the systematic literature review on the topic of transformation rules between elements of UML class diagrams and OWL 2 constructs. The purpose of the analysis is to present the extent to which state-of-the-art transformation rules cover the semantics expressed in class diagrams. On the basis of the analysis, new transformation rules expressing the semantics not yet covered but expected from the point of view of domain modelling pragmatics have been defined.
Results: The first result is the revision and extension of the transformation rules identified in the literature. The second original result is a proposition of verification rules necessary to check if a UML class diagram is compliant with the OWL 2 domain ontology.
Conclusion: The proposed transformations can be used for automatic validation of compliance of UML class diagrams with respect to OWL 2 domain ontologies.

Keywords: UML, OWL 2, transformation rules, verification rules

1. Introduction

In [1], we presented an idea of a method for semantic validation of Unified Modeling Language (UML) class diagrams [2] with the use of OWL 2 Web Ontology Language (OWL 2) [3] domain ontologies. While UML has been known for many years, OWL is a much younger formalism and its main purpose is to represent knowledge in the Semantic Internet. The choice of OWL is justified by the fact that knowledge, and in particular ontologies collected on the Internet, will be increasingly used in business modelling as the first stage of software development. The proposed approach [1] requires a transformation of an UML class diagram constructed by a modeller into its semantically equivalent OWL 2 representation. Despite the fact that there are many publications which define some UML to OWL 2 transformations, to the best of the authors' knowledge, no study has investigated a complete mapping covering all diagram elements emphasized by pragmatic needs. This paper seeks to contribute in this field with a special focus on providing a full transformation of elements of an UML class diagram which are commonly used in business and conceptual modelling. All the transformations described in this paper and the method of validation explained in [1], have been implemented in a tool whose prototype was presented in [4]. On the basis of the proposed UML–OWL transformations, the tool has been further extended. Currently, the tool offers validation of the modified diagram, and can automatically suggest how the diagram

Submitted: 6 June 2018; Revised: 7 October 2018; Accepted: 7 November 2018; Available online: 14 December 2018

should be corrected on the basis of the ontology. A necessary requirement before the UML class diagram can be validated with the use of OWL domain ontology is that the diagram and the ontology must follow one agreed domain vocabulary. Moreover, the domain ontology must be consistent because it is the knowledge base for the area.

Our research is limited to the static elements of UML class diagrams – the behavioural aspect represented by class operations is omitted. This is due to the fact that the semantics of UML operations cannot be effectively expressed with the use of OWL 2 constructs, which do not represent behaviour. In order to identify which transformation rules of UML class diagrams into OWL constructs have already been proposed, we have performed a systematic review of literature. The extracted rules have been analysed, compared and extended. The resulting findings of how to conduct the transformation of UML class diagram to its OWL 2 representation are described further in this paper. In the rest of this paper, OWL always means OWL 2, if not stated otherwise.

Besides the transformation rules, the method of semantic validation of UML class diagrams requires the so called verification rules. This aspect is an original element of this research. Transforming the UML elements to OWL may introduce some new properties that may be in conflict with the ontology. The verification rules are specified in the form of either OWL verification axioms or verification queries.

It is then checked that the verification axioms are not present in the domain ontology because, if they are included, the diagram is contradictory to the domain knowledge. In other words, we can say that the verification axioms detect if the semantics of the diagram transformation is compliant with the axioms included in the domain ontology. Considering the inverse transformation (from the ontology to the diagram), the presence of the verification axioms in the domain ontology means that the reengineering transformation would remain in conflict with the semantics of the UML class diagram. In [1], we presented some examples of the consequences of a reengineering transformation of OWL to UML which does not take into account the verification rules.

Verification queries are used for extracting information from a domain ontology, the kind of information that could not be provided through inspecting the class diagram itself. The domain ontology can have more information regarding the elements of a UML class diagram, which is not explicitly expressed on the diagram. For example, the ontology can contain information about individuals. To give a more detailed perspective, the verification queries are used for: (a) checking if the classes denoted as abstract in the UML class diagram do not have any individuals assigned in the OWL domain ontology, (b) verifying if the multiplicity (of both the attributes and the association ends) is not violated on the side of the OWL domain ontology, and (c) checking if the user-defined list of literals of the specified enumerations on the UML class diagram is compliant with those defined in the OWL domain ontology. Technically, all verification queries are defined with the use of SPARQL[1] language.

The verification of UML class diagrams with the use of the proposed method is possible thanks to the initial normalization of the domain ontology and the normalization of the transformation axioms. The concept of OWL ontology normalization is our proposition [5]. Any input OWL 2 DL ontology after normalization is presented in a new but semantically equivalent form because the normalization rules only change the structure but do not affect the semantics of axioms or expressions in the OWL 2 ontology. The normalized OWL 2 DL ontologies have a unified structure of axioms so that they can be algorithmically compared without the need to conduct additional complex calculations. The extensive details of conducting the transformation of OWL 2 ontologies to their normalized form are described in [5]. In the rest of the paper OWL domain ontology is understood as OWL domain ontology after normalization (it should be done only once). Before comparison of axioms, all transformation axioms are also normalized (tool automatically saves transformation axioms also in the normalized form so that no additional delay is needed in

[1]SPARQL Query Language: https://www.w3.org/TR/sparql11-overview/

the verification algorithm). For the purpose of being compliant with the literature and for the potential use of transformation axioms for other purposes, all transformation axioms presented in this paper are not normalized. On the other hand, due to the fact that verification axioms are our proposition preliminary designed to support verification of UML class diagrams, some rules for verification axioms are already defined in the normalized form in order to reduce the number of unnecessary redundant verifications, the rest rules for verification axioms are not yet normalized for the purpose of clarity for readers but the tool also automatically saves the verification axioms directly as normalized.

In practical use of UML to OWL transformation, the initial phase involving modeller's attention is required. The modeller has to assure that all class attributes and association end names in one UML class are uniquely named. Otherwise, the transformation rules may generate repeating OWL axioms which may lead to inconsistencies or may be semantically incorrect.

The remainder of this article is organized as follows. Section 2 summarizes related works. Section 3 outlines which elements of UML class diagrams are commonly used in business and conceptual modelling. Section 4 describes the process and the results of the conducted systematic literature review which was focused on identifying the state-of-the-art transformation rules for translating UML class diagrams into their OWL representation. Section 5 presents the revised and extended transformation rules and proposes the verification rules. Section 6 summarises some important differences between OWL 2 and UML languages and their impact on transformation. Section 7 illustrates application of transformation and verification rules to example UML class diagrams. Section 8 is dedicated to the tool that implements the transformations. Finally, Section 9 concludes the paper.

2. Class diagrams in business and conceptual modelling

The UML specification [2] does not strictly specify which elements of UML class diagrams should or should not be included in the specific diagrams and this decision is always left to modellers. However, not all model elements are equally useful in the practice of business and conceptual modelling with UML class diagrams.

In [6], it is suggested that a full variety of UML constructs is not needed until the implementation phase and it is practiced that a subset of diagram elements useful for conceptual modelling in the business context is selected. The following static elements of UML class diagrams are suggested in literature as the most important in business and conceptual modelling [7, 8]:

- named classes,
- attributes of classes with types (either primitive or structured datatypes),
- associations between the classes (including aggregation) with the specified multiplicity of the association ends,
- generalization relationships.

Modelling a complex business requires using several views, each of which focuses on a particular aspect of business. Following [7], there are four commonly used Business Views: Business Vision View, Business Process View, Business Structure View and Business Behaviour View. The UML class diagrams are identified as useful [7] in Business Vision View and Business Structure View.

The UML class diagrams in a Business Vision View [7] are used to create conceptual models which establish a common vocabulary and demonstrate relationships among different concepts used in business. The important elements of UML class diagrams in the conceptual modelling are named classes and associations between the classes as they define concepts. The classes can have attributes as well as a textual explanation which together constitute a catalogue of terms. The textual descriptions may not be necessarily visible on the UML diagram but should be retrievable with the help of modelling tools. In the conceptual modelling with UML, attributes and operations of classes are not so much important [7] (can be defined only if needed) but relationships among the classes should be already correctly captured in models.

The UML class diagrams in a Business Structure View [7] are focused on presenting a structure of resources, products, services and information regarding the business including the organization of the company. The class diagrams in this view often include classes containing attributes with types and operations, as well as generalizations and associations with the specified multiplicity.

In [8], modelling business processes with UML class, activity and state machine diagrams is suggested. UML class diagrams with a number of predefined classes are used to describe process entity representatives (activities, agents, resources and artefacts). The examples in [8] present a business process at the level of the UML class diagram as consisting of classes with attributes, class generalizations, associations between the classes (including aggregation) with a specified multiplicity of the association ends. The class attributes are typed with either primitive or structured datatypes.

We have not found further recommendations for using additional static UML class diagram elements in the context of business or conceptual modelling in other reviewed literature positions. If the selected UML class diagram is compliant with the domain, it is reasonable to examine the diagram further. For example, the question outside the scope of this research is about the role of OCL[2] in business and conceptual modelling with UML class diagrams. Some other works investigate this aspect, e.g. in [9] an approach to translate OCL invariants into OWL 2 DL axioms can be found.

3. Review process

Kitchenham and Charters in [10] provide guidelines for performing systematic literature review (SLR) in software engineering. Following [10], a systematic literature review is a means of evaluating and interpreting all available research relevant to a particular research question, and aims at presenting a fair evaluation of a research topic by using a rigorous methodology. This section describes the carried out review aimed at identifying studies describing mappings of UML class diagrams to their OWL representations.

3.1. Research question

The research question is:
RQ: “What transformation rules between elements of UML class diagrams and OWL constructs have already been proposed?”

3.2. Data sources and search queries

In order to make the process repeatable, the details of our search strategy are documented below. The search was conducted in the following online databases: IEEE Xplore Digital Library, Springer Link, ACM Digital Library and Science Direct. These electronic databases were chosen because they are commonly used for searching literature in the field of Software Engineering. Additional searches with the same queries were conducted through ResearchGate and Google scholar in order to discover more relevant publications. These publication channels were searched to find papers published in all the available years until May 2018. The earliest primary study actually included was published in 2006.

For conducting the search, the following keywords were selected: “transformation”, “transforming”, “mapping”, “translation”, “OWL”, “UML” and “class diagram”. The keywords are alternate words and synonyms for the terms used in the research question, which aimed to minimize the effect of differences in terminologies. Pilot searches showed that the above keywords were too general and the results were too broad. Therefore, in order to obtain more relevant results, the search queries were based on the Boolean AND to join terms:

– “transformation” AND “OWL” AND “UML”,
– “transforming” AND “OWL” AND “UML”,
– “mapping” AND “OWL” AND “UML”,
– “translation” AND “OWL” AND “UML”,
– “transformation” AND “OWL” AND “class diagram”,
– “transforming” AND “OWL” AND “class diagram”,

[2]Object Constraint Language (OCL): http://www.omg.org/spec/OCL/

– “mapping” AND “OWL” AND “class diagram”,
– “translation” AND “OWL” AND “class diagram”.

3.3. Inclusion and exclusion criteria

The main inclusion criterion was that a paper provides some transformation rules between UML class diagrams and OWL constructs. Additionally, the study had to be written in English and be fully accessible through the selected online libraries. Additionally, there was a criterion for excluding a paper from the review results if the study described transformation rules between other types of UML diagrams to OWL representation or described transformation rules to other ontological languages.

3.4. Study quality assessment

The final acceptance of the literature was done by applying the quality criteria. The criteria were assigned a binary “yes”/“no” answer. In order for a work to be selected, it needed to provide “yes” answer to both questions from the checklist:

1. Are the transformation rules explicitly defined? For example, a paper could be excluded if it only reported on a possibility of specifying transformation rules for the selected UML elements, but such transformations were not provided.
2. Do the proposed transformation rules preserve the semantics of the UML elements? For example, a paper (or some selected transformation rules within the paper) could be excluded if the proposed rules in the transformation to OWL 2 did not preserve the semantics of the UML elements.

3.5. Study selection

During the search, the candidate papers for full text reading were identified by evaluating their titles and abstracts. The literature was included or excluded based on the selection criteria. The goal was to obtain the literature that answered the research question. The candidate papers, after eliminating duplicates, were fully read. After positive assessment of the quality of the literature items, they were added to the results of the systematic literature review.

Next, if the paper was included, its reference list was additionally scanned in order to identify potential further relevant papers (backward search). Later, the paper selection has additionally been extended by forward search related to works citing the included papers. In both backward search and forward search the papers for full text reading were identified based on reading title and abstract.

3.6. Threats to validity

We have identified threats to the validity of the conducted SLR, grouped in accordance with the categories presented in [11]. Wherever applicable, we included the applied mitigating factors corresponding to the identified threats.

Construct Validity: The specified search queries may not be able to completely cover all adequate search terms related to the research topic. As a mitigating factor, we used alternate words and synonyms for the terms used in the research question.

Internal Validity: The identified treats to internal validity relate to search strategy and further steps of conducting the SLR, such as selection strategy and quality assessment:

1. A threat to validity was caused by lack of assurance that all papers relevant to answering the research question were actually found. A mitigating factor to this threat was conducting a search with several search queries and analyzing the references of the primary studies with the aim of identifying further relevant studies.
2. Another threat was posed by the selected research databases. The threat was reduced by conducting the search with the use of six different electronic databases.
3. A threat was caused by the fact that one researcher conducted SLR. A mitigating factor to the search process and the study selection process was that the whole search process was twice reconducted in April 2018 and May 2018.

The additional procedures did not change the identified studies.

External Validity: External validity concentrates on the generalization of findings derived from the primary studies. The carried search was aimed at identifying transformation rules of elements of UML class diagram to their OWL 2 representation. Some transformation rules could be formulated analogically in some other ontological languages, e.g. DAML+OIL, etc. Similarly, some transformation rules could be formulated analogically in some modelling languages or notations different then UML class diagrams, e.g. in Entity Relationship Diagram (ERD), EXPRESS-G graphical notation for information models, etc. A generalization of findings is out of scope of this research.

Conclusion Validity: The search process was twice reconducted and the obtained results have not changed. However, non-determinism of some database search engines is a threat to the reliability of this and any other systematic review because the literature collected through non-deterministic search engines might not be repeatable by other researchers with exactly the same results. In particular it applies to the results obtained with the use of Google scholar and ResearchGate.

4. Related work

4.1. Search results

In total, the systematic literature review identified 18 studies. 14 literature positions were found during the search: [12–26]. Additional 30 studies were excluded based on the quality assessment exclusion criterion.

Additional 3 studies were obtained through the analysis of the references of the identified studies (the backward search): [27–29].

The forward search has not resulted in any paper included. The majority of papers had already been examined during the main search and had already been either previously included or excluded. In the forward search, three papers describing transformation rules have been excluded because they were not related to UML. Most other papers have been excluded because they have not described transformation rules. Two papers have been excluded because the transformation rules were only mentioned but not defined. A relatively large number (approximately 20%) of articles has been excluded based on the language criterion – they had not been written in English (the examples of the observed repetitive languages: Russian, French, Turkish, Chinese, and Spanish).

The results of the search with respect to data sources are as follows (data source – number of selected studies): ResearchGate – 6; Springer Link – 3; IEEE Xplore Digital Library – 2; Google Scholar – 2; ACM Digital Library – 1; Science Direct – 1. In order to eliminate duplicates that were found in more than one electronic database, the place where a paper was first found was recorded.

Table 1. Search results versus years of publication

Year of publication	Resulting papers
2006	[23]
2008	[14, 16, 21, 27]
2009	[13]
2010	[26]
2012	[12, 17, 20, 22, 25]
2013	[19, 24, 28]
2014	[29]
2015	[18]
2016	[15]

To summarize, the identified studies include: 3 book chapters, 8 papers published in journals, 5 papers published in the proceedings of conferences, 1 paper published in the proceedings of a workshop and 1 technical report. The identified primary studies were published in the years between 2006–2016 (see Table 1). What can be observed is that the topic has been gaining greater attention since 2008. It should not be a surprise because OWL became a formal W3C recommendation in 2004.

4.2. Summary of identified literature

Most of the identified studies described just a few commonly used diagram elements (i.e. UML class, binary association and generalization between the classes or associations) while some other diagram elements obtained less attention in the literature (i.e. multiplicity of attributes, n-ary association or generalization sets). For some class diagram elements the literature offers incomplete transformations. Some of the transformation rules defined in the selected papers are excluded from the findings based on the quality criteria defined in Section 3.4. The state-of-the-art transformation rules were revised and extended. Section 5 contains detailed references to the literature related to relevant transformations. The following is a short description of the included studies:

The work presented in [18] transforms into OWL some selected elements of UML models containing multiple UML class, object and statechart diagrams in order to analyze consistency of the models. A similar approach is presented in [19], which is focused on detecting inconsistency in models containing UML class and statechart diagrams.

The papers [15, 17, 29] investigate the differences and similarities between UML and OWL in order to present transformations of selected (and identified as useful) elements of UML class diagram. In [29], the need for UML–OWL transformation is additionally motivated by not repeating the modelling independently in both languages.

In [14], a possible translation of few selected elements of several UML diagrams to OWL is presented. The paper takes into account a set of UML diagrams: use case, package, class, object, timing, sequence, interaction overview and component. The behavioural elements in UML diagrams in [14] are proposed to be translated to OWL with annotations.

The work of [26] focuses on representing UML and MOF-like metamodels with the use of OWL 2 language. The approach includes proposition of transforming Classes and Properties.

The paper [27] compares OWL abstract syntax elements to the equivalent UML features and appropriate OCL statements. The analysis is conducted in the direction from OWL to UML. For every OWL construct its UML interpretation is proposed.

The article [20] describes transformation rules for UML data types and class stereotypes selected from UML profile defined in ISO 19103. A full transformation for three stereotypes is proposed. The article describes also some additional OWL–UML mappings. The focus of [28] is narrowed to transformation of data types only.

Some works are focused on UML–OWL transformations against the single application domain. The paper [21] depicts the applicability of OWL and UML in the modelling of disaster management processes. In [16], transportation data models are outlined and the translation of UML model into its OWL representation is conducted for the purpose of reasoning.

The works presented in [12, 13, 23] are focused on extracting ontological knowledge from UML class diagrams and describe some UML–OWL mappings with the aim to reuse the existing UML models and stream the building of OWL domain ontologies. The paper [12] from 2012 extends and enhances the conference paper [13] from 2009. Both papers were analysed during the process of collecting the data in case of detection of any significant differences in the description of transformation rules.

In [22], UML classes are translated into OWL. Finally, [24, 25] present a few transformations of class diagram elements to OWL.

5. UML class diagram and its OWL 2 representation

This section presents **transformation rules (TR)** which seek to transform the elements of UML class diagrams to their equivalent representations expressed in OWL 2. Some of the transformation rules come from the literature identified in the review (e.g. **TR1** in Table 2). Another group of rules have their archetypes in the state-of-the-art transformation rules but we have refined them in order to clarify their contexts of use (e.g. $\mathbf{TR_A}$, $\mathbf{TR_C}$ in Section 6.2), or extend their application to a broader scope (e.g. **TR1** in Table 5). The remaining transformation rules are our new propositions (e.g. **TR5** in Table 7).

In contrast to the approaches available in the literature, together with the transformation rules we define the **verification rules (VR)** for all elements of a UML class diagram wherever applicable. The need for specifying verification rules results from the fact that we would like to check the compliance of the OWL representation of UML class diagram with the OWL domain ontology. The role of verification rules is to detect if the semantics of a diagram is not in conflict with the knowledge included in the domain ontology.

All the transformation and verification rules are presented in Tables 2–21. We took into consideration all the static elements of UML class diagrams, which are important from the point of view of pragmatics (see Section 2). To summarize the results, most of the UML elements which are recommended [7, 8] in business or conceptual modelling with UML class diagrams are fully transformable to OWL 2 constructs:

– *Class* (Table 2),
– attributes of the *Class* (Table 4),
– multiplicity of the attributes (Table 5),
– binary *Association* – both between two different *Classes* (Table 6) as well as from a *Class* to itself (Table 7),
– multiplicity of the *Association* ends (Table 9),
– *Generalization* between *Classes* (Table 12),
– *Integer*, *Boolean* and *UnlimitedNatural* primitive types (Table 18),
– structured *DataType* (Table 19),
– *Enumeration* (Table 20),
– *Comments* to the *Class* (Table 21),

We additionally fully translated into OWL 2 the following UML elements which have not been identified among recommended for business or conceptual modelling but can be used in further stages of software development:

– *Generalization* between *Associations* (Table 13),
– *GeneralizationSet* with constraints (Tables 14–17),
– *AssociationClass* (Table 10 and Table 11),

The UML and OWL languages have different expressing power. We consider also the partial transformation, which is possible for:

– *String* and *Real* primitive types because they have only similar but not equivalent to OWL 2 types (Table 18),
– aggregation and composition can be transformed only as simple associations (Tables 6 and 7),
– n-ary *Association* – OWL 2 offers only binary relations, a pattern to mitigate the problem of transforming n-ary *Association* is presented (Table 8),
– *AbstractClass* – OWL 2 does not offer any axiom for specifying that a class must not contain any individuals. Although, it is impossible to confirm that the UML abstract class is correctly defined with respect to the OWL 2 domain ontology, it can be detected if it is not (Table 3).

The tables below present for each UML element its short description, a graphical symbol, transformation rules, verification rules, explanations or comments, limitations of the transformations (if any) and the works related for the transformation rules (if any). Additionally, some tables include references to Section 7, where examples of UML–OWL transformations are presented.

The convention for transformation and verification rules presentation is semi-formal, similar to the convention used in other publication presenting transformation rules, e.g. [17, 20]. It seems to be more readable than a strict formal presentation. However, a formal presentation is implicitly defined in the programming tool which transforms any UML class diagram into a set of OWL axioms.

All OWL 2 constructs are written with the use of a functional-style syntax [30]. Additionally, the following convention is used:

– C – indicates an OWL class;
– CE (possibly with an index) – indicates a class expression;
– OPE (possibly with an index) – indicates an object property expression;
– DPE (possibly with an index) – indicates a data property expression;
– $\alpha = \beta$ – means textual identity of α and β OWL 2 constructs;

- $\alpha \neq \beta$ – means textual difference of α and β OWL 2 constructs;
- The elements of UML meta-model, UML model, and OWL entities or literals named in the UML model are written with the use of *italic* font;
- The OWL 2 constructs (axioms, expressions and datatypes) and SPARQL queries are written in **bold**.

All presented SPARQL queries use the following prefixes:

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://*...selected ontology*>

5.1. Transformation of UML classes with attributes

Table 2. Classes and the defined rules

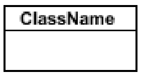
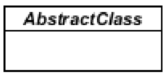
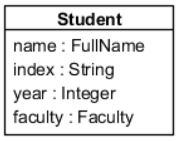
UML element	*Class*
Description of UML element	In UML, a *Class* [2] is purposed to specify a classification of objects.
Symbol of UML element	
Transformation rules	**TR1**: Specify declaration axiom for UML *Class* as OWL **Class**: **Declaration**(**Class**(:*ClassName*))
Verification rules	**VR1**: Check if :*ClassName* class has the **HasKey** axiom defined in the domain ontology. **HasKey**(:*ClassName*(OPE$_1$.. OPE$_m$) (DPE$_1$.. DPE$_n$))
Comments to the rules	1. Regarding **VR1**: The OWL **HasKey** axiom assures [30, 31] that if two named instances of a class expression contain the same values of all object and data property expressions, then these two instances are the same. This axiom is in contradiction with the semantics of UML class because UML specification allows for creating different objects with exactly the same properties.
Related works	In [12–23, 25–27], UML class is transformed to OWL with the use of **TR1** axiom.
Example	Section 7 example 1, 2 and 3

Table 3. Abstract classes and the defined rules

UML element	Abstract *Class*
Description of UML element	In UML, an abstract *Class* [2] cannot have any instances and only its subclasses can be instantiated.
Symbol of UML element	
Transformation rules	**Not possible.** The UML abstract classes cannot be translated into OWL because OWL does not offer any axiom for specifying that a class must not contain any individuals.
Verification rules	**VR1**: Check if the domain ontology contains any individual specified for the :*AbstractClass*. **SELECT** (**COUNT** (**DISTINCT** ?**ind**) **as** ?**count**) **WHERE** {?**ind rdf:type** :*AbstractClass*}

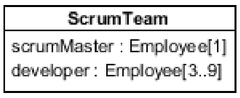
	If the :*AbstractClass* does not contain any individual specified in the domain ontology, the SPARQL query returns zero: "0"^^<http://www.w3.org/2001/XMLSchema#**integer**>
Comments to the rule	OWL follows the Open World Assumption [30], therefore, even if the ontology does not contain any instances for a specific class, it is unknown whether the class has any instances. We cannot confirm that the UML abstract class is correctly defined with respect to the OWL domain ontology, but we can detect if it is not (**VR1** checks if the class specified as abstract in the UML class diagram is indeed abstract in the domain ontology).
Related works	In [17, 20, 29], UML abstract class is stated as not transformable into OWL. In [17, 20], it is proposed that **DisjointUnion** is used as an axiom which covers some semantics of UML abstract class. However, UML specification does not require an abstract class to be a union of disjoint classes, and the **DisjointUnion** axiom does not prohibit creating members of the abstract superclass, therefore, it is insufficient.

Table 4. Attributes and the defined rules

UML element	Attributes
Description of UML element	The UML attributes [2] are *Properties* that are owned by a *Classifier*, e.g. *Class.*
Symbol of UML element	 Student name : FullName index : String year : Integer faculty : Faculty
Transformation rules	**TR1**: Specify declaration axiom(s) for attribute(s) as OWL data or object properties respectively **Declaration**(**ObjectProperty**(:*name*)) **Declaration**(**DataProperty**(:*index*)) **Declaration**(**DataProperty**(:*year*)) **Declaration**(**ObjectProperty**(:*faculty*)) **TR2**: Specify data (or object) property domains for attribute(s) **ObjectPropertyDomain**(:*name* :*Student*) **DataPropertyDomain**(:*index* :*Student*) **DataPropertyDomain**(:*year* :*Student*) **ObjectPropertyDomain**(:*faculty* :*Student*) **TR3**: Specify data (or object) property ranges for attribute(s) (for transformation of UML *PrimitiveTypes* refer to Table 18, for transformation of UML structure*DataTypes* to Table 19) **ObjectPropertyRange**(:*name* :*FullName*) **DataPropertyRange**(:*index* **xsd**:**string**) **DataPropertyRange**(:*year* **xsd**:**integer**) **ObjectPropertyRange**(:*faculty* :*Faculty*)
Verification rules	**VR1**: Check if the domain ontology contains **ObjectPropertyDomain** (or **DataPropertyDomain)** axiom specified for OPE (or DPE) where CE is specified for a different than given UML *Class* (here :*Student*) **ObjectPropertyDomain**(:*name* CE), where CE $\neq$:*Student* **DataPropertyDomain**(:*index* CE), where CE $\neq$:*Student* **DataPropertyDomain**(:*year* CE), where CE $\neq$:*Student* **ObjectPropertyDomain**(:*faculty* CE), where CE $\neq$:*Student* **VR2**: Check if the domain ontology contains **ObjectPropertyRange** (or **DataPropertyRange)** axiom specified for OPE (or DPE) where CE is

	specified for a different than given UML structure*DataType* (or DR is specified for a different than given UML *PrimitiveType*) **ObjectPropertyRange**(:*faculty* CE), where CE ≠ :*Faculty* **DataPropertyRange**(:*index* DR), where DR ≠ **xsd:string** **DataPropertyRange**(:*year* DR), where DR ≠ **xsd:integer** **ObjectPropertyRange**(:*name* CE), where CE ≠ :*FullName*
Comments to the rules	1. Both UML attributes and associations are represented by one meta-model element – *Property.* OWL also allows one to define properties. A transformation of UML attribute to OWL data property or OWL object property bases on its type. If the type of the attribute is *PrimitiveType* it should be transformed into OWL **DataProperty**. However, if the type of the attribute is a structured *DataType*,it should be transformed into an OWL **ObjectProperty**. 2. **VR1** checks whether or not the object properties (or respectively data properties) indicate that the UML attributes are specified for given UML *Class.* 3. **VR2** checks whether or not the object properties (or respectively data properties) indicate that the UML attributes of the specified UML *Class* have specified given types, either *PrimitiveTypes* or structured *DataTypes.*
Related works	**TR1**–**TR3** are proposed in [15–17, 20]. In [12–14, 18, 19, 21–24], all UML attributes are translated into data properties only.
Example	Section 7 example 2 and 3

Table 5. Multiplicity of attributes and the defined rules

UML element	Multiplicity of attributes
Description of UML element	In [2], multiplicity bounds of*MultiplicityElement* are specified in the form of <*lower-bound*> ".." <*upper-bound*>. The *lower-bound* is of a non-negative *Integer* type and the *upper-bound* is of an *UnlimitedNatural* type. The strictly compliant specification of UML in version 2.5 defines only a single value range for *MultiplicityElement.* However, in practical examples it is sometimes useful not limit oneself to a single interval. Therefore, the below UML to OWL mapping covers a wider case – a possibility of specifying more value ranges for a multiplicity element. Nevertheless, if the reader would like to strictly follow the current UML specification, the particular single *lower..upper* bound interval is therein also comprised. In comparison to UML, the OWL specification [30] defines three class expressions **ObjectMinCardinality**, **ObjectMaxCardinality** and **ObjectExactCardinality** for specifying the individuals that are connected by an object property to at least, at most or exactly to a given number (non-negative integer) of instances of the specified class expression. Analogically, **DataMinCardinality**, **DataMaxCardinality** and **DataExactCardinality** class expressions are used for data properties.
Symbol of UML element	 ScrumTeam scrumMaster : Employee[1] developer : Employee[3..9]
Transformation rules	**TR1**: If UML attribute is specified with the use of OWL **ObjectProperty**, its multiplicity should be specified analogously to **TR1** from Table 9 (multiplicity of association ends). If UML attribute is specified with the use of OWL **DataProperty**, its multiplicity should be specified with the use of axiom: **SubClassOf**(:*ClassName multiplicityExpression*) We define *multiplicityExpression* as one of class expressions: **A**, **B**, **C** or **D**: **A**. a **DataExactCardinality** class expression if UML *MultiplicityElement* has *lower-bound* equal to its *upper-bound*, e.g. "1..1", which is semantically equivalent to "1".

B. a **DataMinCardinality** class expression if UML *MultiplicityElement* has *lower-bound* of *Integer* type and *upper-bound* of unlimited *upper-bound*, e.g. “2..*”.
C. an **ObjectIntersectionOf** class expression consisting of **DataMinCardinality** and **DataMaxCardinality** class expressions if UML *MultiplicityElement* has *lower-bound* of *Integer* type and *upper-bound* of *Integer* type, e.g. “4..6”.
D. an **ObjectUnionOf** class expression consisting of a combination of **ObjectIntersectionOf** class expressions (if needed) or **DataExactCardinality** class expressions (if needed) or one **DataMinCardinality** class expression (if the last range has unlimited *upper-bound*), if UML *MultiplicityElement* has more value ranges specified, e.g. “2, 4..6, 8..9, 15..*”.
The following is the result of application of **TR1** to the above diagram:

SubClassOf(:*ScrumTeam*
  ObjectExactCardinality(1 :*scrumMaster* :*Employee*))
SubClassOf(:*ScrumTeam* **ObjectIntersectionOf**(
  ObjectMinCardinality(3 :*developer* :*Employee*)
  ObjectMaxCardinality(9 :*developer* :*Employee*)))

Verification rules

VR1: Regardless of whether or not the UML attribute is specified with the use of OWL **DataProperty** or **ObjectProperty**, the verification rule is defined with the use of the SPARQL query (only applicable for multiplicities with maximal *upper-bound* not equal “*”).

SELECT ?**vioInd** (**count** (?**range**) **as** ?n)
WHERE {?**vioInd** :*leaf* ?**range** } **GROUP BY** ?**vioInd**
HAVING (?n > *maxUpperBoundValue*)

where *maxUpperBoundValue* is a maximal *upper-bound* value of the multiplicity range. If the query returns a number greater than 0, it means that UML multiplicity is in contradiction with the domain ontology (**?vioInd** lists individuals that cause the violation).
The following is the result of definition of **VR1** to the above diagram:
maxUpperBoundValue for *scrumMaster*: 1
SPARQL query for *scrumMaster*:

SELECT ?**vioInd** (**count** (?**range**) **as** ?n)
WHERE { ?**vioInd** : *scrumMaster* ?**range** } **GROUP BY** ?**vioInd**
HAVING (?n > 1)

maxUpperBoundValue for *developer*: 9
SPARQL query for *developer*:

SELECT ?**vioInd** (**count** (?**range**) **as** ?n)
WHERE { ?**vioInd** : *developer* ?**range** } **GROUP BY** ?**vioInd**
HAVING (?n > 9)

VR2: Check if the domain ontology contains **SubClassOf** axiom, which specifies CE with different multiplicity of attributes than it is derived from the UML class diagram.
SubClassOf(:*ScrumTeam* CE)

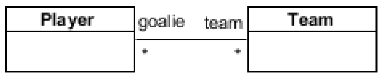
Comments to the rules

1.It should be noted that *upper-bound* of UML *MultiplicityElement* can be specified as unlimited: “*”. In OWL, cardinality expressions serve to restrict the number of individuals that are connected by an object property expression to a given number of instances of a specified class expression [30]. Therefore, UML unlimited *upper-bound* does not add any information to OWL ontology, hence it is not transformed.
2. Regarding **TR1**: the rule relies on the **SubClassOf(** CE$_1$ CE$_2$) axiom, which restricts CE$_1$ to necessarily inherit all the characteristics of CE$_2$, but not the other way around. The difference of using **EquivalentClasses(** CE$_1$ CE$_2$)

	axiom is that the relationship is implied to go in both directions (and the reasoner would infer in both directions). 3. Regarding **VR1**: As motivated in [17], reasoners that base on Open World Assumption can detect a violation of an upper limit of the cardinality restrictions only. This is caused by the fact that in Open World Assumption it is assumed that there might be other individuals beyond those that are already presented in the ontology. The verification rules for the cardinality expressions are defined with the use of SPARQL queries, which are aimed to verify whether or not the domain ontology does have any individuals that are contradictory to **TR1** axiom. Therefore, the **VR1** verifies the existence of individuals that are connected to the selected object property a number of times that is greater than the specified UML multiplicity. 4. The rule **VR2** verifies if the ontology contains axioms which describe multiplicity of *Attributes* different than the multiplicity specified in the UML class diagram.
Related works	The related works present only partial solutions for multiplicity of attributes. In [29], a solution for a single value interval is proposed. In [17], multiplicity associated with class attributes is transformed to a single expression of exact, maximum or minimum cardinality. In [24], multiplicity is transformed only into maximum or minimum cardinality.
Example	Section 7 example 2

5.2. Transformation of UML associations

Table 6. Binary Associations between two different Classes and the defined rules

UML element	Binary *Association* (between two different *Classes*)
Description of UML element	Following [2], a binary *Association* specifies a semantic relationship between two *memberEnds* represented by *Properties*. Please note that in accordance with specification [2], the association end names are not obligatory. In the method of validation and the prototype tool we followed the same convention which is adopted for all metamodel diagrams throughout the specification ([2, page 61]): *If an association end is unlabeled, the default name for that end is the name of the class to which the end is attached, modified such that the first letter is a lowercase letter.* Due to the fact that our method of transformation requires additionally unique names, either the modeller has to rename the names, or the tool in such cases automatically adds subsequent numbers to the names. For transformation of UML multiplicity of the association ends, refer to Table 9.
Symbol of UML element	
Transformation rules	**TR1**: Specify declaration axiom(s) for object properties **Declaration**(**ObjectProperty**(:*team*)) **Declaration**(**ObjectProperty**(:*goalie*)) **TR2**: Specify object property domains for association ends (note: if the association contains an *AssociationClass*, the domains should be transformed in accordance with **TR1** from Table 10) **ObjectPropertyDomain**(:*team* :*Player*) **ObjectPropertyDomain**(:*goalie* :*Team*) **TR3**: Specify object property ranges for association ends **ObjectPropertyRange**(:*team* :*Team*) **ObjectPropertyRange**(:*goalie* :*Player*)

	TR4: Specify **InverseObjectProperties** axiom for the association **InverseObjectProperties**(:*team* :*goalie*)
Verification rules	**VR1**: Check if **AsymmetricObjectProperty** axiom is specified for any of UML association ends. **AsymmetricObjectProperty**(:*goalie*) **AsymmetricObjectProperty**(:*team*) **VR2**: Check if the domain ontology contains **ObjectPropertyDomain** specified for the same OPE but different CE than it is derived from the UML class diagram. **ObjectPropertyDomain**(:*team* CE), where CE$\neq$:*Player* **ObjectPropertyDomain**(:*goalie* CE), where CE$\neq$:*Team* **VR3**: Check if the domain ontology contains **ObjectPropertyRange** axiom specified for the given OPE but different CE than it is derived from the UML class diagram. **ObjectPropertyRange**(:*team* CE), where CE$\neq$:*Team* **ObjectPropertyRange**(:*goalie* CE), where CE$\neq$:*Player*
Comments to the rules	1. **TR4** is specified to state that both resulting object properties are part of one UML *Association.* 2. Regarding **VR1**: A binary *Association* between two different *Classes* may not be asymmetric. Please refer to Table 7 for explanation of asymmetric binary *Association* from a *Class* to itself. 3. Regarding **VR2**: If the domain ontology contains **ObjectPropertyDomain** specified for the same OPE but different CE than it is derived from the UML class diagram, the Association is defined in the ontology but between different Classes. 4. Regarding **VR3**: If the domain ontology contains **ObjectPropertyRange** axiom specified for the given OPE but different CE than it is derived from the UML class diagram, the Association is defined in the ontology but between different Classes.
Limitations of the mapping	1. UML *Association* has two important aspects. The first is related to its existence and it can be transformed to OWL. It should be noted that UML introduces an additional notation related to communication between objects. The second one concerns navigability of the association ends which is untranslatable because OWL does not offer any equivalent concept. 2. Both UML aggregation and composition can be only transformed to OWL as regular *Associations.* This approach loses the specific semantics related to the composition or aggregation, which is untranslatable to OWL.
Related works	In [14–22, 25, 27], **TR1**–**TR3** rules for the transformation of UML binary association to object property domain and range are proposed. In [15, 20, 26], **TR4** rule is additionally proposed. In [17, 20], a unidirectional association is transformed into one object property and a bi-directional association into two object properties (one for each direction). This interpretation does not seem to be sufficient because if an association end is not navigable in UML 2.5, access from the other end may be possible, but it might not be efficient ([2, page 198]).
Example	Section 7 example 1 and 3

Table 7. Binary Association from the Class to itself and the defined rules

UML element	Binary *Association* from a *Class* to itself
Description of UML element	A binary *Association* [2] contains two *memberEnds* represented by *Properties.* For transformation of multiplicity of the association ends, refer to Table 9.

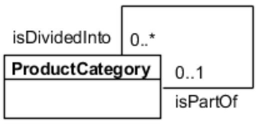
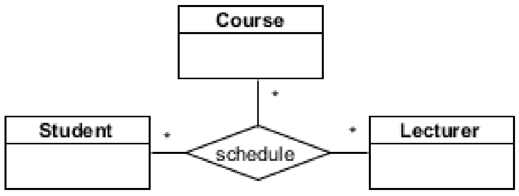
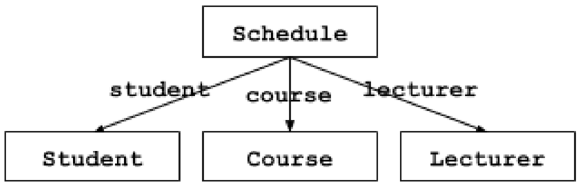
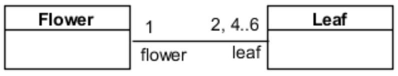
Symbol of UML element	 isDividedInto 0..* ProductCategory 0..1 isPartOf
Transformation rules	**TR1–TR4**: The same as **TR1–TR4** from Table 6. **TR5**: Specify **AsymmetricObjectProperty** axiom for each UML association end **AsymmetricObjectProperty**(:*isPartOf*) **AsymmetricObjectProperty**(:*isDividedInto*)
Verification rules	**VR1** is the same as **VR2** from Table 6. **VR2** is the same as **VR3** from 6.
Comments to the rules	1. In **TR2** domain and range of binary association is the same UML class. **VR4** checks if the domain ontology does not specify a different domain or range for the *Association*. 2. In **TR5** object property OPE is defined as asymmetric. In OWL, if an individual x is connected by OPE to an individually, then y cannot be connected by OPE to x.
Limitations of the mapping	The same as presented in Table 6.
Related works	For **TR1–TR4** related works are analogous as in Table 6, while **TR5** is our new proposition. In [15], the UML binary association from the *Class* to itself is converted to OWL with the use of two **ReflexiveObjectProperty** axioms. We do not share this approach because a specific association may be reflexive but in the general case it is not true. The **ReflexiveObjectProperty** axiom states that each individual is connected by OPE to itself. In consequence, it would mean that every object of the class should be connected to itself. The UML binary *Association* has a different meaning where the association ends have different roles.
Example	Section 7 example 2

Table 8. *N*-ary associations and the defined rules

UML element	*N*-ary *Association*
Description of UML element	UML *n*-ary *Association* [2] specifies the relationship between three or more *memberEnds* represented by *Properties*. For transformation of UML multiplicity of the association ends refer to Table 9.
Symbol of UML element	
Transformation rules	**Not possible** to directly represent UML *n*-ary associations in OWL 2. The following is a partial transformation based on the pattern presented in [32]. The pattern requires creating a new class and N new properties to represent the *n*-ary association. The figure below shows the corresponding classes and properties.  Schedule student course lecturer Student Course Lecturer

	TR1: Specify declaration axiom for the new class which represent the *n*-ary association (declaration axioms for other classes are added following Table 2) **Declaration**(**Class**(:*Schedule*)) **TR2**: Specify declaration axiom(s) for object properties **Declaration**(**ObjectProperty**(:*student*)) **Declaration**(**ObjectProperty**(:*course*)) **Declaration**(**ObjectProperty**(:*lecturer*)) **TR3**: Specify object property domains for association ends **ObjectPropertyDomain**(:*student* :*Student*) **ObjectPropertyDomain**(:*course* :*Course*) **ObjectPropertyDomain**(:*lecturer* :*Lecturer*) **TR4**: Specify object property ranges for association ends **ObjectPropertyRange**(:*student* :*Schedule*) **ObjectPropertyRange**(:*course* :*Schedule*) **ObjectPropertyRange(** :*lecturer* :*Schedule*) **TR5**: Specify **SubClassOf(** CE_1**ObjectSomeValuesFrom(** OPE CE_2)) axioms, where CE_1 is a newly added class, OPE are properties representing the UML *Association* and CE_2 are corresponding UML *Classes* **SubClassOf**(:*Schedule* ObjectSomeValuesFrom(:*student* :*Student*)) **SubClassOf**(:*Schedule* ObjectSomeValuesFrom(:*course* :*Course*)) **SubClassOf**(:*Schedule* ObjectSomeValuesFrom(:*lecturer* :*Lecturer*))
Verification rules	None
Limitations of the mapping	Properties in OWL 2 are only binary relations. Three solutions to represent an n-ary relation in OWL are presented in W3C Working Group Note [32] in a form of ontology patterns. Among the proposed solutions for n-ary association, we selected one the most appropriate to UML and we supplemented it by adding unlimited "*" multiplicity at every association end of the UML *n*-ary association.
Related works	The transformation rules (**TR1, TR2, TR5**) of a *n*-ary association base on the pattern proposed in [32]. **TR3, TR4** complement the rules, analogically as it is in binary associations. In [15], a partial transformation for *n*-ary association is proposed, but one rule should be modified because an object property expression is used in the place of a class expression.

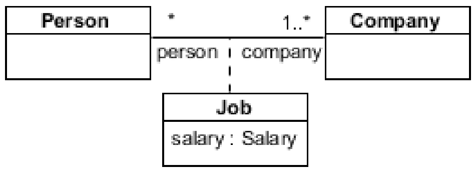
Table 9. Multiplicity of association ends and the defined rules

UML element	Multiplicity of *Association* ends
Description of UML element	Description of multiplicity is presented in Table 5 (multiplicity of attributes). If no multiplicity of association end is defined, the UML specification implies a multiplicity of 1.
Symbol of UML element	
Transformation rules	**TR1**: For each association end with the multiplicity different than "*" specify axiom: **SubClassOf**(:*ClassName multiplicityExpression*) We define *multiplicityExpression* as one of class expressions: **A**, **B**, **C** or **D**: **A.** an **ObjectExactCardinality** if UML *MultiplicityElement* has *lower-bound* equal to its *upper-bound*, e.g. "1..1", which is semantically equivalent to "1". **B.** an **ObjectMinCardinality** class expression if UML *MultiplicityElement* has *lower-bound* of *Integer* type and *upper-bound* of unlimited *upper-bound*, e.g. "2..*". **C.** an **ObjectIntersectionOf** consisting of **ObjectMinCardinality** and **ObjectMaxCardinality** class expressions if UML *MultiplicityElement* has *lower-bound* of *Integer* type and *upper-bound* of *Integer* type, e.g. "4..6".

	D. an **ObjectUnionOf** consisting of a combination of **ObjectIntersectionOf** class expressions (if needed) **OR ObjectExactCardinality** class expressions (if needed) **OR** one **ObjectMinCardinality** class expression (if the last range has an unlimited *upper-bound*), if UML *MultiplicityElement* has more value ranges specified, e.g. "2, 4..6, 8..9, 15..*". The following is a result of application of **TR1** to the above diagram: **SubClassOf**(:*Leaf* **ObjectExactCardinality**(1 :*flower* :*Flower*)) **SubClassOf**(:*Flower* **ObjectUnionOf**(**ObjectExactCardinality**(2 :*leaf* :*Leaf*) **ObjectIntersectionOf**(**ObjectMinCardinality**(4 :*leaf* :*Leaf*) **ObjectMaxCardinality**(6 :*leaf* :*Leaf*)))) **TR2**: Specify **FunctionalObjectProperty** axiom if a multiplicity of the association end equals 1. **FunctionalObjectProperty**(:*flower*)
Verification rules	**VR1:** The rule is defined with the use of the SPARQL query (only applicable for multiplicities with maximal *upper-bound* not equal "*"). **SELECT** ?**vioInd** (**count** (?**range**) **as** ?n) **WHERE** { ?**vioInd** :*leaf* ?**range** } **GROUP BY** ?**vioInd** **HAVING** (?n > *maxUpperBoundValue*) where *maxUpperBoundValue* is a maximal *upper-bound* value of the multiplicity range. If the query returns a number greater than 0, it means that UML multiplicity is in contradiction with the domain ontology (**?vioInd** lists individuals that cause the violation). The following is a result of application of **VR1** to the above diagram: *maxUpperBoundValue* for *flower*: 1 SPARQL query for *flower*: **SELECT** ?**vioInd** (**count** (?**range**) **as** ?n) **WHERE** { ?**vioInd** :*flower* ?**range** } **GROUP BY** ?**vioInd** **HAVING** (?n > 1) *maxUpperBoundValue* for *leaf*: 6 SPARQL query for *leaf*: **SELECT** ?**vioInd** (**count** (?**range**) **as** ?n) **WHERE** { ?**vioInd** :*leaf* ?**range** } **GROUP BY** ?**vioInd** **HAVING** (?n > 6) **VR2**: Check if the domain ontology contains **SubClassOf** axiom, which specifies CE with different multiplicity of association ends than is derived from the UML class diagram. **SubClassOf**(:*Leaf* CE) **SubClassOf**(:*Flower* CE)
Comments to the rules	1. The **TR1**, **TR2** and **VR1** rules are explained in Table 5. 2. Regarding **TR2**: The **FunctionalObjectProperty** axiom states that each individual can have a maximum of one outgoing connection of the specified object property expression. 3. The rule **VR2** verifies whether or not the ontology contains axioms, which describe multiplicity of association ends different than multiplicity specified in the UML class diagram. 4. We have considered one additional validation rule for checking if the domain ontology contains **FunctionalObjectProperty** axiom specified for the association end which multiplicity is different from 1: **FunctionalObjectProperty**(:*leaf*) However, after analyzing of this rule, it would never be triggered. This is caused by the fact that the violation of cardinality is checked by **TR1** rule. And specifying **FunctionalObjectProperty** axiom in the ontology along with the transformation axiom describing cardinality different than 1, makes the ontology inconsistent.

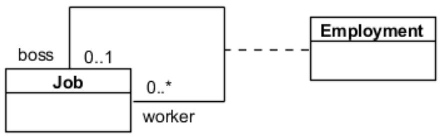
Related works	The related works present partial solutions for multiplicity of association ends. In [14, 18, 19, 26], the multiplicity of an association end is mapped to **SubClassOf** axiom containing a single **ObjectMinCardinality** or **ObjectMaxCardinality** class expression. In [17], **ObjectExactCardinality** expression is also considered and **TR2** rule is additionally proposed. In [12, 13, 15, 21, 22, 24], multiplicity is only suggested to be transformed into OWL cardinality restrictions.
Example	Section 7 example 1, 2 and 3

Table 10. Association class (the association is between two different classes) and the defined rules

UML element	*AssociationClass* (the *Association* is between two different *Classes*)
Description of UML element	*AssociationClass* [2] is both an *Association* and a *Class*, and preserves the semantics of both. Table 11 presents *AssociationClass* in the case when association is from a UML *Class* to itself.
Symbol of UML element	
Transformation rules	The binary association between *Person* and *Company* UML classes should be transformed to OWL in accordance with the transformations **TR1**, **TR3**–**TR4** from Table 6. The object property ranges should be specified in accordance with **TR2** from Table 6. The transformation of object property domains between *Person* and *Company* UML classes should be transformed with **TR1** rule below. Transformation of multiplicity of the association ends are specified in Table 9. The attributes of the UML association class :*Job* should be specified in accordance with the transformation rules presented in Table 4. If multiplicity of attributes is specified, it should be transformed in accordance with the guidelines from Table 5. **TR1**: Specify object property domains for *Association* ends **ObjectPropertyDomain**(:*person* **ObjectUnionOf**(:*Company* :*Job*)) **ObjectPropertyDomain**(:*company* **ObjectUnionOf**(:*Person* :*Job*)) **TR2**: Specify declaration axiom for UML association class as OWL **Class**: **Declaration**(**Class**(:*Job*)) **TR3**: Specify declaration axiom for object property of UML *AssociationClass* **Declaration**(**ObjectProperty**(:*job*)) **TR4**: Specify object property domain for UML *AssociationClass* **ObjectPropertyDomain**(:*job* **ObjectUnionOf**(:*Person* :*Company*)) **TR5**: Specify object property range for UML association class **ObjectPropertyRange**(:*job* :*Job*)
Verification rules	**VR1**: Check if :*Job* class has the **HasKey** axiom defined in the domain ontology. **HasKey**(:*Job* ($\mathrm{OPE}_1 \ldots \mathrm{OPE}_m$) ($\mathrm{DPE}_1 \ldots \mathrm{DPE}_n$)) **VR2**: Check if the domain ontology contains **ObjectPropertyDomain** axiom specified for a given OPE (from *Association* ends and *AssociationClass*) but different CE than is derived from the UML class diagram. **ObjectPropertyDomain**(:*person*CE), where CE $\neq$ **ObjectUnionOf**(:*Company* :*Job*), **ObjectPropertyDomain**(:*company* CE), where CE $\neq$ **ObjectUnionOf**(:*Person* :*Job*) **ObjectPropertyDomain**(:*job* CE), where CE $\neq$ **ObjectUnionOf**(:*Person* :*Company*)

	VR3: Check if the domain ontology contains **ObjectPropertyRange** axiom specified for the same object property of UML association class but different CE than it is derived from the UML class diagram. **ObjectPropertyRange**(:*job* CE), where CE ≠ :*Job*
Comments to the rules	1. The proposed transformation of UML association class covers both the semantics of the UML class (**TR1**–**TR2**, plus the transformation of attributes possibly with multiplicity), as well as UML *Association* (**TR3**–**TR5**, plus the transformation of multiplicity of *Association* ends). 2. Regarding **TR1** and **TR3**: The domain of the specified property is restricted to those individuals that belong to the union of two classes. 3. Explanation of **VR1** is analogous to **VR1** from Table 2. 4. **VR2** checks if the UML *Association* and *AssociationClass* is specified correctly with respect to the domain ontology. **VR3** checks if the domain ontology does not specify a different range for the *AssociationClass*.
Related works	**TR1, TR3**–**TR5** transformation rules of the UML association class to OWL are original propositions and the proposed transformations to OWL cover full semantics of the UML *AssociationClass*. The literature [14, 15, 25] present only partial solutions for transforming UML association classes. In [14], it is only suggested that UML *AssociationClass* be transformed with the use of the named class (here: *Job*) and two functional properties that demonstrate associations (here: *Job–Person* and *Job–Company*). In [15, 25] some rules are with an unclear notation, more precisely *AssociationClass* is transformed to OWL with the use of **TR2** rule and a set of mappings which base on a specific naming convention.
Example	Section 7 example 3

Table 11. Association class (the Association is from a UML Class to itself) and the defined rules

UML element	*AssociationClass* (the *Association* is from a UML *Class* to itself)
Description of UML element	*AssociationClass* [2] is both an *Association* and a *Class*, and preserves the semantics of both. Table 10 presents *AssociationClass* in the case when association is between two different classes.
Symbol of UML element	
Transformation rules	All comments presented in Table 10 in TR section are applicable also for *AssociationClass* where association is from a UML *Class* to itself. Additionally, **TR5** from Table 7 has to be specified. Transformation rules **TR1**, **TR2**, **TR3** and **TR5** are the same as **TR1**, **TR2**, **TR3** and **TR5** from Table 10. Except for **TR4**, which has form: **TR4**: Specify object property domain for UML *AssociationClass* **ObjectPropertyDomain**(:*employment* :*Job*)
Verification rules	**VR1** and **VR3**: The same as **VR1** and **VR3** from Table 10. **VR2**: Check if the domain ontology contains **ObjectPropertyDomain** axiom specified for a given OPE (from *Association* ends and *AssociationClass*) but different CE than is derived from the UML class diagram. **ObjectPropertyDomain**(:*boss* CE), where CE ≠ **ObjectUnionOf**(:*Job* :*Employment*), **ObjectPropertyDomain**(:*worker* CE), where CE ≠ **ObjectUnionOf**(:*Job* :*Employment*) **ObjectPropertyDomain**(:*employment* CE), where CE ≠ :*Job*

Comments to the rules	The same as presented in Table 10.
Related works	The same as presented in Table 10.

5.3. Transformation of UML generalization relationship

Table 12. Generalization between classes and the defined rules

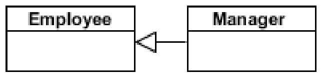
UML element	*Generalization* between *Classes*
Description of UML element	*Generalization* [2] defines specialization relationship between *Classifiers*. In case of UML classes it relates a more specific *Class* to a more general *Class*.
Symbol of UML element	
Transformation rules	**TR1**: Specify **SubClassOf(** CE_1 CE_2) axiom for the generalization between UML classes, where CE_1 represents a more specific and CE_2 a more general UML *Class*. **SubClassOf**(:*Manager* :*Employee*)
Verification rules	**VR1**: Check if the domain ontology contains **SubClassOf(** CE_2 CE_1) axiom specified for classes, which take part in the generalization relationship, where CE_1 represents a more specific and CE_2 a more general UML *Class*. **SubClassOf**(:*Employee* :*Manager*)
Related works	In [15, 17–19, 21–23, 25–27, 29] **TR1** is specified. In [12, 13], generalizations are only suggested to be transformed to OWL with the use of **SubClassOf** axiom.
Example	Section 7 example 1 and 2.

Table 13. Generalization between associations and the defined rules

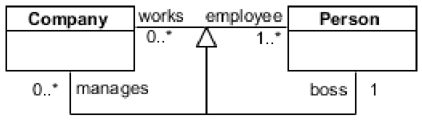
UML element	*Generalization* between *Associations*
Description of UML element	*Generalization* [2] defines specialization relationship between *Classifiers*. In case of the UML associations it relates a more specific *Association* to more general *Association*.
Symbol of UML element	 Company works employee Person 0..* 1..* 0..* manages boss 1
Transformation rules	**TR1**: Specify two **SubObjectPropertyOf(** OPE_1 OPE_2) axioms for the generalization between UML *Association*, where OPE_1 represents a more specific and OPE_2 a more general association end connected to the same UML *Class*. **SubObjectPropertyOf**(:*manages* :*works*) **SubObjectPropertyOf**(:*boss* :*employee*)
Verification rules	**VR1**: Check if the domain ontology contains **SubObjectPropertyOf(** OPE_2 OPE_1) axiom specified for associations, which take part in the generalization relationship, where OPE_1 represents a more specific and OPE_2 a more general UML association end connected to the same UML *Class*. **SubObjectPropertyOf**(:*works* :*manages*) **SubObjectPropertyOf**(:*employee* :*boss*)
Related works	In [15, 17, 18, 26, 27, 29], **TR1** rule is proposed additionally with two **InverseObjectProperties** axioms (one for each association). This table does not add a transformation rule for **InverseObjectPropertie** axioms because the axioms were already added while transforming binary associations (see Tables 6, 7.
Example	Section 7 example 1

Table 14. GeneralizationSet with {incomplete, disjoint} constraints and the defined rules

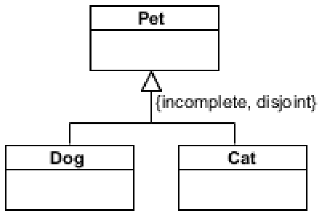
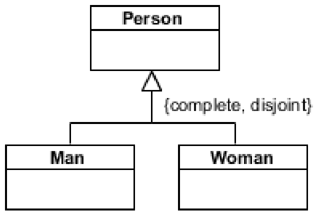
UML element	*GeneralizationSet* with {*incomplete, disjoint*} constraints
Description of UML element	UML *GeneralizationSet* [2] groups generalizations; *incomplete* and *disjoint* constraints indicate that the set is not complete and its specific *Classes* have no common instances.
Symbol of UML element	
Transformation rules	**TR1**: Specify **DisjointClasses** axiom for every pair of more specific *Classes* in the *Generalization.* **DisjointClasses**(:*Dog* :*Cat*)
Verification rules	**VR1**: Check if the domain ontology contains any of **SubClassOf(** CE_1 CE_2) or **SubClassOf(** CE_2 CE_1) axioms specified for any pair of more specific *Classes* in the *Generalization.* **SubClassOf**(:*Dog* :*Cat*) **SubClassOf**(:*Cat* :*Dog*)
Comments to the rules	1. **TR** and **VR** for *Generalization* between UML *Classes* are specified in Table 12. 2. Regarding **TR1**: the **DisjointClasses(** CE_1 CE_2) axiom states that no individual can be at the same time an instance of both CE_1 and CE_2 for $\mathrm{CE}_1 \neq \mathrm{CE}_2$.
Related works	In [15, 17, 29], **TR1** rule is proposed.

Table 15. GeneralizationSet with {complete, disjoint} constraints and the defined rules

UML element	*GeneralizationSet* with {*complete, disjoint*} constraints
Description of UML element	UML *GeneralizationSet* [2] is used to group generalizations; *complete* and *disjoint* constraints indicate that the generalization set is complete and its specific *Classes* have no common instances.
Symbol of UML element	
Transformation rules	**TR1**: Specify **DisjointUnion** axiom for UML *Classes* within the *GeneralizationSet.* **DisjointUnion**(:*Person* :*Man* :*Woman*)
Verification rules	**VR1**: Check if the domain ontology contains **SubClassOf(** CE_1 CE_2) or **SubClassOf(** CE_2 CE_1) axioms specified for any pair of more specific *Classes* in the *Generalization.* **SubClassOf**(:*Man* :*Woman*) **SubClassOf**(:*Woman* :*Man*) **VR2**: Check if the domain ontology contains **DisjointUnion(** C CE_1.. CE_N) axiom specified for the given more general UML *Class* (here :*Person*) and at least one more specific UML *Class* different than those specified on the UML class diagram.

	DisjointUnion(:*Person* $CE_1 .. CE_N$)
Comments to the rules	1.**TR** and **VR** for *Generalization* between UML *Classes* are specified in Table 12. 2. **VR2** checks if the *GeneralizationSet* with {*complete, disjoint*} constraints is defined correctly with respect to domain ontology.
Related works	In [15, 17, 29], **TR1** is proposed.
Example	Section 7 example 2

Table 16. GeneralizationSet with {incomplete, overlapping} constraints and the defined rules

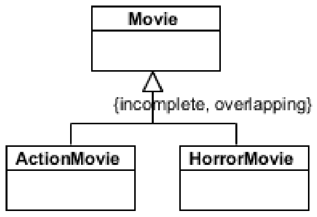
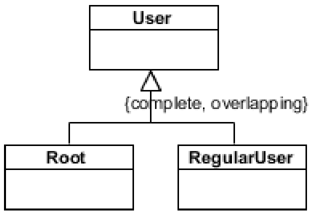
UML element	*GeneralizationSet* with {*incomplete, overlapping*} constraints
Description of UML element	UML *GeneralizationSet* [2] is used to group generalizations; *incomplete* and *overlapping* constraints indicate that the generalization set is not complete and its specific *Classes* do share common instances. If no constraints of *GeneralizationSet* are specified, {*incomplete, overlapping*} are assigned as default values ([2, p. 119]).
Symbol of UML element	 Movie {incomplete, overlapping} ActionMovie HorrorMovie
Transformation rules	None
Verification rules	**VR1**: Check if the domain ontology contains **DisjointClasses(** CE_1 CE_2) axiom specified for any pair of more specific *Classes* in the *Generalization.* **DisjointClasses**(:*ActionMovie* :*HorrorMovie*)
Comments to the rules	1. **TR** and **VR** for *Generalization* between UML *Classes* are specified in Table 12. 2. OWL follows Open World Assumption and by default incomplete knowledge is assumed, hence the UML *incomplete* and *overlapping* constraints of *GeneralizationSet* do not add any new knowledge to the ontology, so no **TR** are specified. 3. UML *overlapping* constraint states that specific UML *Classes* in the *Generalization* do share common instances. Therefore, the **DisjointClasses** axiom is a verification rule **VR1** for the constraint (the axiom assures that no individual can be at the same time an instance of both classes).
Related works	None

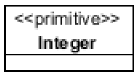
Table 17. GeneralizationSet with {complete, overlapping} constraints and the defined rules

UML element	*GeneralizationSet* with {*complete, overlapping*} constraints
Description of UML element	UML *GeneralizationSet* [2] is used to group generalizations; *complete* and *overlapping* constraints indicate that the generalization set is complete and its specific *Classes* do share common instances.

Symbol of UML element	
Transformation rules	**TR1**: Specify **EquivalentClasses** axiom for UML *Classes* within the *GeneralizationSet*. **EquivalentClasses**(:*User* **ObjectUnionOf**(:*Root* :*RegularUser*))
Verification rules	**VR1**: Check if the domain ontology contains **DisjointClasses(** CE_1 CE_2) axiom specified for any pair of more specific *Classes* in the *Generalization*. **DisjointClasses**(:*Root* :*RegularUser*) **VR2**: Check if the domain ontology contains **EquivalentClasses** axiom specified for the given more general UML *Class* (here :*User*) and **ObjectUnionOf** containing at least one UML *Class* different than specified on the UML class diagram for the more specific classes. **EquivalentClasses**(:*User* **ObjectUnionOf**($\mathrm{CE}_1..\mathrm{CE}_N$)), where **ObjectUnionOf**($\mathrm{CE}_1..\mathrm{CE}_N$) $\neq$ **ObjectUnionOf**(:*Root* :*RegularUser*)
Comments to the rules	1. **TR** and **VR** for *Generalization* between UML *Classes* are specified in Table 12. 2. Explanation for **VR1** is presented in Table 16. 3. **VR2** checks if the *GeneralizationSet* with {*complete, overlapping*} constraint is compliant with the domain ontology.
Related works	In [15], **TR1** rule is defined with additional **DisjointClasses(** :*Dog* :*Cat*) axiom. However, the **DisjointClasses** axiom should not be specified for the UML *Classes* which may share common instances.

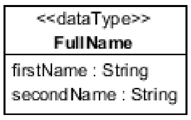
5.4. Transformation of UML data types

Table 18. Primitive types and the defined rules

UML element	*PrimitiveType*
Description of UML element	The UML *PrimitiveType* [2] defines a predefined *DataType* without any substructure. The UML specification [2] predefines five primitive types: *String*, *Integer*, *Boolean*, *UnlimitedNatural* and *Real*.
Symbol of UML element	
Transformation rules	It is impossible to define unambiguously the transformation of UML String and UML Real type, therefore, the decision on the final transformation is left to the modeller. The proposed transformations for the two types base on their similarity in UML 2.5 and OWL 2 languages. The transformation between UML predefined primitive types and OWL 2 datatypes: **TR1**: UML *String* has only a similar OWL 2 type: **xsd:string** String types in the sense of UML and OWL are countable sets. It is possible to define an infinite number of equivalence functions, which is left to the user, wherein, the UML is imprecise as to what the accepted characters are. **TR2**: UML *Integer* has an equivalent OWL 2 type: **xsd:integer** **TR3**: UML *Boolean* has an equivalent OWL 2 type: **xsd:boolean** **TR4**: UML *Real* has two similar OWL 2 types: **xsd:float** and **xsd:double** Both UML and OWL 2 languages describe types that are subsets of the set of

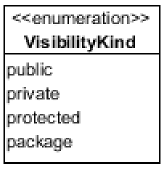
	real numbers. The subsets are countable. If one accepts a 32 or 64-bit precision of UML *Real* type, they will obtain an appropriate compatibility with OWL 2 **xsd:float** or **xsd:double** types. **TR5**: UML *UnlimitedNatural* can be explicitly defined in OWL 2 as: **DatatypeDefinition**(:*UnlimitedNatural* **DataUnionOf**(**xsd**:nonNegativeInteger **DataOneOf**("∗"^^**xsd**:**string**)))
Verification rules	None
Comments to the rules	The UML specification [2] on page 717 defines the semantics of five predefined *PrimitiveTypes*. The specification of OWL 2 [30] also offers predefined datatypes (many more than UML). **TR1**: An instance of UML *String* [2] defines a sequence of characters. Character sets may include non-Roman alphabets. On the other hand, OWL 2 supports **xsd:string** defined in XML Schema [33]. The value space of **xsd:string** [33] is a set of finite-length sequences of zero or more characters that match the Char production from XML, where Char is any Unicode character, excluding the surrogate blocks, FFFE, and FFFF. The cardinality of **xsd:string** is defined as countably infinite. Due to the fact that the ranges of characters differ, UML *String* and OWL 2 **xsd:string** are only similar datatypes. **TR2**: An instance of UML *Integer* [2] is a value in the infinite set of integers $(\ldots, -2, -1, 0, 1, 2, \ldots)$. OWL 2 supports **xsd:integer** defined in XML Schema [33]. The value space of **xsd:integer** is an infinite set $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$. The cardinality is defined as countably infinite. The UML *Integer* and OWL 2 **xsd:integer** types can be seen as equivalent. **TR3**: An instance of UML *Boolean* [2] is one of the predefined values: true and false. OWL 2 supports **xsd:boolean** defined in XML Schema [33], which represents the values of two-valued logic :{true, false}. The lexical space of **xsd:boolean** is a set of four literals: 'true', 'false', '1' and '0' but the lexical mapping for **xsd:boolean** returns true for 'true' or '1', and false for 'false' or '0'. Therefore the UML *Boolean* and **xsd:boolean** types can be seen as equivalent. **TR4**: An instance of UML *Real* [2] is a value in the infinite set of real numbers. Typically [2] an implementation will internally represent *Real* numbers using a floating point standard such as ISO/IEC/IEEE 60559:2011, whose content is identical [2] to the predecessor IEEE 754 standard. On the other hand, OWL 2 supports **xsd:float** and **xsd:double**, which are defined in XML Schema [33]. The **xsd:float** [33] is patterned after the IEEE single-precision 32-bit floating point datatype IEEE 754-2008 and the **xsd:double** [33] after the IEEE double-precision 64-bit floating point datatype IEEE 754-2008. The value space contains the non-zero numbers $m \times 2^e$, where m is an integer whose absolute value is less than 2^{53} for **xsd:double** (or less than 2^{24} for **xsd:float**), and e is an integer between -1074 and 971 for **xsd:double** (or between -149 and 104 for **xsd:float**), inclusive. Due to the fact that the value spaces differ, UML *Real* and OWL 2 **xsd:double** (or **xsd:float**) are only similar datatypes. **TR5**: An instance of UML *UnlimitedNatural* [2] is a value in the infinite set of natural numbers (0, 1, 2...) plus unlimited. The value of unlimited is shown using an asterisk ('*'). UnlimitedNatural values are typically used [2] to denote the *upper-bound* of a range, such as a multiplicity; unlimited is used whenever the range is specified as having no *upper-bound*. The UML *UnlimitedNatural* can be defined in OWL and added to the ontology as a new datatype (**TR5**).
Related works	The related works are not precise with respect to the transformation of primitive types. In [17, 27–29], some mappings of UML and OWL types are only mentioned.
Example	Section 7 example 2

Table 19. Structured data types and the defined rules

UML element	Structured *DataType*
Description of UML element	The UML structured *DataType* [2] has attributes and is used to define complex data types.
Symbol of UML element	 <<dataType>> FullName firstName : String secondName : String
Transformation rules	**TR1**: Specify declaration axiom for UML data type as OWL class: **Declaration**(**Class**(:*FullName*)) **TR2**: Specify declaration axiom(s) for attributes – as OWL data or object properties respectively (see Table 4 for more information regarding attributes) **Declaration**(**DataProperty**(:*firstName*)) **Declaration**(**DataProperty**(:*secondName*)) **TR3**: Specify data (or object) property domains for attributes **DataPropertyDomain**(:*firstName* :*FullName*) **DataPropertyDomain**(:*secondName* :*FullName*) **TR4**: Specify data (or object) property ranges for attributes (OWL 2 datatypes for UML primitive types are defined in Table 18) **DataPropertyRange**(:*firstName* **xsd**:**string**) **DataPropertyRange**(:*secondName* **xsd**:**string**) **TR5**: Specify **HasKey** axiom for the UML data type expressed in OWL with the use of a class uniquely identified by the data and/or object properties. **HasKey**(:*FullName* ( ) (:*firstName* :*secondName*))
Verification rules	**VR1**: Check if the domain ontology contains **DataPropertyDomain** axiom specified for DPE where CE is different than given UML structured *DataType* **DataPropertyDomain**(:*firstName* CE), where CE$\neq$:*FullName* **DataPropertyDomain**(:*secondName* CE), where CE$\neq$:*FullName* **VR2**: Check if the domain ontology contains **DataPropertyRange** axiom specified for DPE where CE is different than given UML *PrimitiveType* **DataPropertyRange**(:*firstName* DR), where DR$\neq$**xsd**:**string** **DataPropertyRange**(:*secondName* DR), where DR$\neq$**xsd**:**string**
Comments to the rules	1. UML *DataType* [2] is a kind of *Classifier*, whose instances are identified only by their values. All instances of a UML *DataType* with the same value are considered to be equal [2]. A similar meaning can be assured in OWL with the use of **HasKey** axiom. The **HasKey** axiom [30] assures that each instance of the class expression is uniquely identified by the object and/or data property expressions. 2. **VR1** checks whether the data properties indicate that the UML attributes are correct for the specified UML structured *DataType*. 3. **VR2** checks whether the data properties indicate that the UML attributes of the specified UML structured *DataType* have correctly specified *PrimitiveTypes*.
Limitations of the mapping	Due to the fact that we define the UML structure *DataType* as an OWL **Class** and not as an OWL **Datatype** (see Section 6.3 for further explanation), the presented transformation results in some consequences. A limitation is posed by the fact that the instances of the UML *DataType* are identified only by their value [2], while the **TR1** rule opens a possibility of explicitly defining the named instances for the **Entity** in OWL.
Related works	In [28, 29] **TR1**–**TR5** rules and in [15] **TR2**–**TR5** rules are proposed for the transformation of UML structured *DataType*. In [17], it is only noted that UML *DataTypes* can be defined in OWL with the use of **DatatypeDefinition** axiom

	but no example is provided. The related works specify exclusively the data properties as attributes of the structured data types in **TR2**. We extend the state-of-the-art **TR2** transformation rule by the possibility of defining also object properties, wherever needed (see Table 4).
Example	Section 7 example 2

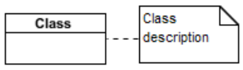
Table 20. Enumeration and the defined rules

UML element	*Enumeration*
Description of UML element	UML *Enumerations* [2] are kinds of *DataTypes*, whose values correspond to one of user-defined literals.
Symbol of UML element	 <<enumeration>> VisibilityKind public private protected package
Transformation rules	**TR1**: Specify declaration axiom for UML *Enumeration* as OWL **Datatype**: **Declaration**(**Datatype**(:*VisibilityKind*)) **TR2**: Specify **DatatypeDefinition** axiom including the named **Datatype** (here :*VisibilityKind*) with a data range in a form of a predefined enumeration of literals (**DataOneOf**). **DatatypeDefinition**(:*VisibilityKind* **DataOneOf**(*"public" "private" "protected" "package"*))
Verification rule	**VR1**: Check if the list of user-defined literals in the *Enumeration* on the class diagram is correct and complete with respect to the OWL datatype definition for *:VisibilityKind* included in the domain ontology. The SPARQL query: **SELECT** ?**literal** { :*VisibilityKind* **owl**:**equivalentClass** ?**dt**. ?**dt a rdfs**:**Datatype** ; **owl**:**oneOf**/**rdf**:**rest**∗/**rdf**:**first** ?**literal** } returns a list of literals of the enumeration from the domain ontology. The list of literals should be compared with the list of user-defined literals on the class diagram if the UML *Enumeration* includes a correct and complete list of literals.
Limitations of the mapping	*Enumerations* [2] in UML are specializations of a *Classifier* and therefore can participate in generalization relationships. OWL has no construct allowing for generalization of datatypes. See Section 6.3 for further explanation.
Related works	In [17, 20, 28, 29], UML *Enumeration* is transformed to OWL with the use of **TR1**–**TR2** rules.

5.5. Transformation of UML comments

Table 21. Comment and the defined rules

UML element	*Comment* to the *Class*
Description of UML element	In accordance with [2], every kind of UML *Element* may own *Comments* which add no semantics but may represent information useful to the reader. In OWL it is possible to define the annotation axiom for OWL **Class**, **Datatype**, **ObjectProperty**, **DataProperty**, **AnnotationProperty** and **NamedIndividual**. The textual explanation added to UML *Class* is identified as useful for conceptual modelling [7], therefore the *Comments* that are connected to UML *Classes* are taken into consideration in the transformation.

Symbol of UML element	
Transformation rules	**TR1**: Specify annotation axiom for UML *Comment* **AnnotationAssertion**(**rdfs**:**comment** :*Class* " *Class description* "^^**xsd**:**string**)
Verification rule	Not applicable
Comments to the rule	As UML *Comments* add no semantics, they are not used in any method of semantic validation [1]. In OWL the **AnnotationAssertion** [30] axiom does not add any semantics either, and it only improves readability.
Related works	The transformation of UML *Comments* in the context of mapping to OWL has not been found in literature.

6. Influence of UML–OWL differences on transformations

Obviously, OWL 2 and UML 2.5 languages differ from each other.

In general, notice that OWL ontologies are based on the Open World Assumption while UML class diagrams are based on Closed World Assumption. We can compare a UML class diagram to a given OWL ontology assuming that this ontology is in a given state. Examining that the UML class diagram conforms to the OWL ontology we transform the diagram into equivalent OWL representation and check if this representation forms a subset of the ontology. So, the notion of semantic equivalence relates only to the UML class diagram and its OWL representation.

The further part of the section focuses exclusively on two selected differences which influence the form of transformations.

6.1. Instances

OWL 2 defines several kinds of axioms: declarations, axioms about classes, axioms about objects and data properties, datatype definitions, keys, assertions (used to state that individuals are instances of, e.g. class expressions) and axioms about annotations. What can be observed is that the information about classes and their instances (in OWL called individuals) coexists within a single ontology.

On the other hand, in UML two different kinds of diagrams are used in order to present the classes and objects. UML defines object diagrams which represent instances of class diagrams at a certain moment in time. The object diagrams focus on presenting attributes of objects and relationships between objects.

Despite the fact that information about the objects is not present in UML class diagrams, verification rules in the form of SPARQL queries take advantage of the knowledge about individuals in the domain ontology. The rules are useful in validation of class diagrams against the selected domain ontologies as they can check, for example, if an abstract class is indeed abstract (does not have any direct instances in ontology) or if multiplicity restrictions are specified correctly.

6.2. Disjointness in OWL 2 and UML

In OWL 2 an individual can be an instance of several classes [34]. It is also possible to state that no individual can be an instance of selected classes, which is called class disjointness. The information that some specific classes are disjoint is part of domain knowledge which serves a purpose of reasoning.

OWL specification emphasises [34]: *In practice, disjointness statements are often forgotten or neglected. The arguable reason for this could be that intuitively, classes are considered disjoint unless there is other evidence. By omitting disjointness statements, many potentially useful consequences can get lost.*

What can be observed in typical existing OWL ontologies, axioms of disjointness (**DisjointClasses**, **DisjointObjectProperties** and **DisjointDataProperties**) are stated for classes, object properties or data properties only for the most evident situations. If disjointness is not specified, the semantics of OWL states that the ontology does not contain enough information that disjointness takes place. For example, it is possible that the information is actually true but it has not been included in the ontology.

On the other hand, in a UML class diagram every pair of UML classes (which are not within one generalization set with an *overlapping* constraint) is disjoint, where disjointness is understood in the way that the classes have no common instances. This aspect of UML semantics could be mapped to OWL with the use of an extensive set of additional transformations. The transformations would not be intuitive from the perspective of OWL and should add a lot of unnecessary information which might never be useful due to the fact that, e.g. one should consider every pair of classes on the diagram and add additional axioms for it.

For the purpose of completeness of our revision, below we present transformation rules also for disjointness:

– *Transformation rule for disjointness of UML classes* ($\mathbf{TR_A}$): Specify **DisjointClasses** axiom for every pair of UML Classes: CE_1, CE_2 where $CE_1 \neq CE_2$ and the pair is not in the generalization relation or within one generalization set with an overlapping constraint. *Comment*: The $\mathbf{TR_A}$ rule for classes within a generalization relationship was originally proposed in [17, 18, 20]. We have refined the rule in order to cover only the pairs of classes which are not only in a direct generalization relation but also not within one *GeneralizationSet* with an overlapping constraint. This is caused by the fact that the *GeneralizationSet* with the *overlapping* constraint (see Tables 16 and 17) defines specific *Classes*, which do share common instances. Please note that UML *GeneralizationSet* with *disjoint* constraint adds **DisjointClasses** axioms – either directly or indirectly through **DisjointUnion** axiom (see Tables 14 and 15).
– *Transformation rule for disjointness of UML attributes* ($\mathbf{TR_B}$): Specify **DisjointObjectProperties** axiom for every pair OPE_1, OPE_2 where $OPE_1 \neq OPE_2$ of object properties within the same UML Class (domain of both OPE_1 and OPE_2 is the same OWL **Class**) and specify **DisjointDataProperties** axiom for every pair DPE_1, DPE_2 where $DPE_1 \neq DPE_2$ of object properties within the same UML Class (domain of both DPE_1 and DPE_2 is the same OWL **Class**).
Comment: The $\mathbf{TR_B}$ rule is original proposition.
– *Transformation rule for disjointness of UML associations* ($\mathbf{TR_C}$): Specify **DisjointObjectProperties** axiom for every pair of association ends OPE_1 and OPE_2 where $OPE_1 \neq OPE_2$ and OPE_1 is not generalized by OPE_2 and OPE_2 is not generalized by OPE_1 and domain and range of OPE_1 and OPE_2 are the same classes.
Comment: In [17, 20], it is suggested that **DisjointObjectProperties** and **DisjointDataProperties** axioms for all properties that are not in a generalization relationship should be specified. In a general case this suggestion is not clear, but we have modified the rule to be applicable for UML associations which are not in generalization relationship. Even though the $\mathbf{TR_A}$, $\mathbf{TR_B}$ and $\mathbf{TR_C}$ rules are reasonable from the point of view of covering semantics of a class diagram to OWL, they have not been implemented in a tool for validation of UML class diagram [4] due to their questionable usefulness from the perspective of pragmatics. This is caused by the fact that including these rules would lead to a large increase in the number of axioms in the ontology, which would increase the computational complexity.

6.3. Concepts of class and datatype in UML and OWL

OWL 2 allows specifying declaration axioms for datatypes:

Declaration(**Datatype**(:*DatatypeName*))

However, the current specification of OWL 2 [30] does not offer any constructs neither to specify the internal structure of the datatypes, nor the possibility to define generalization relationships between the datatypes. Both are available in UML 2.5.

Please note that the OWL **HasKey**, **DataPropertyDomain** and **ObjectPropertyDomain** axioms can only be defined for the class expressions (not for the data ranges). Therefore the **TR2–TR5** rules in Table 19 can only be specified if the UML structured *DataType* is declared as an OWL **Class**. This transformation has its consequences, which are presented in Table 19.

If future extensions of the OWL language allow one to precisely define the internal structure of datatypes, by analogy, as it is possible in UML, the proposed transformation of UML structured *DataType* presented in Table 19 should then be modified. Additionally, if future extensions of the OWL language allow one to define generalization relationships between datatypes, the currently valid limitation of the transformation of UML *Enumeration* presented in Table 20 will no longer be applicable.

7. Examples of UML–OWL transformations

This section presents some examples of transformations of UML class diagrams to their equivalent OWL representations. The UML class diagram examples are relatively small but cover a number of different UML elements. For clarity of reading, the examples include references to tables from Section 5.

The order of transformations is arbitrary (the resulting set of axioms will always be the same despite the order) but we suggest to conduct the transformations starting from Table 2 to Table 21. In this way, all the classes with attributes will be mapped to OWL first, then the associations and generalization relationships and finally data types and comments.

Each example includes two tables containing transformational and verificational part of UML class diagram (e.g. in Example 1 there are two tables: 22 and 23). Each verificational part should be considered in the context of the selected domain ontology. The Table 23 which presents verificational part of the diagram from Example 1 has been supplemented with additional comments of how each verificational axiom or verificational query should be interpreted. The comments and the ontological background presented in Table 23 is also applicable to other examples.

Example 1

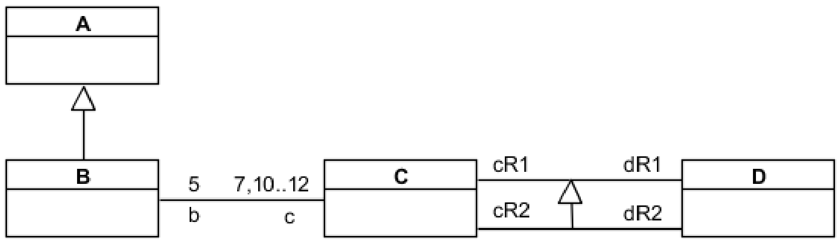


Figure 1. Example 1 of UML class diagram (see Tables 22, 23)

Table 22. Transformational part of UML class diagram from Example 1

Set of transformation axioms	Transformation rules
Transformation of UML Classes	
Declaration(**Class**(:A)) **Declaration**(**Class**(:B)) **Declaration**(**Class**(:C)) **Declaration**(**Class**(:D))	Table 2 **TR1**
Transformation of UML binary Associations between two different Classes	
Declaration(**ObjectProperty**(:b)) **Declaration**(**ObjectProperty**(:c)) **Declaration**(**ObjectProperty**(:$cR1$)) **Declaration**(**ObjectProperty**(:$dR1$)) **Declaration**(**ObjectProperty**(:$cR2$)) **Declaration**(**ObjectProperty**(:$dR2$))	Table 6 **TR1**
ObjectPropertyDomain(:b :C) **ObjectPropertyDomain**(:c :B) **ObjectPropertyDomain**(:$cR1$:D) **ObjectPropertyDomain**(:$dR1$:C) **ObjectPropertyDomain**(:$cR2$:D) **ObjectPropertyDomain**(:$dR2$:C)	Table 6 **TR2**
ObjectPropertyRange(:b :B) **ObjectPropertyRange**(:c :C) **ObjectPropertyRange**(:$cR1$:C) **ObjectPropertyRange**(:$dR1$:D) **ObjectPropertyRange**(:$cR2$:C) **ObjectPropertyRange**(:$dR2$:D)	Table 6 **TR3**
InverseObjectProperties(:b :c) **InverseObjectProperties**(:$cR1$:$dR1$) **InverseObjectProperties**(:$cR2$:$dR2$)	Table 6 **TR4**
Transformation of UML multiplicity of Association ends	
SubClassOf(:C **ObjectExactCardinality**(5 :b :B)) **SubClassOf**(:B **ObjectUnionOf**(**ObjectExactCardinality**(7 :c :C) **ObjectIntersectionOf**(**ObjectMinCardinality**(10 :c :C) **ObjectMaxCardinality**(12 :c :C)))) **SubClassOf**(:C **ObjectExactCardinality**(1 :$dR1$:D)) **SubClassOf**(:D **ObjectExactCardinality**(1 :$cR1$:C)) **SubClassOf**(:C **ObjectExactCardinality**(1 :$dR2$:D)) **SubClassOf**(:D **ObjectExactCardinality**(1 :$cR2$:C))	Table 9 **TR1**
FunctionalObjectProperty(:$dR1$) **FunctionalObjectProperty**(:$cR1$) **FunctionalObjectProperty**(:$dR2$) **FunctionalObjectProperty**(:$cR2$)	Table 9 **TR2**
Transformation of UML Generalization between Classes	
SubClassOf(:B :A)	Table 12 **TR1**
Transformation of UML Generalization between Associations	
SubObjectPropertyOf(:$cR2$:$cR1$) **SubObjectPropertyOf**(:$dR2$:$dR1$)	Table 13 **TR1**

Table 23. Verificational part of UML class diagram from Example 1

Verificational part of UML class diagram	Verification rules
Transformation of UML Classes	
If the domain ontology contains any **HasKey** axiom with any internal structure ($\text{OPE}_1, \ldots$, $\text{DPE}_1 \ldots$) defined for :*A*, :*B*, :*C* or :*D* UML *Class*, the element should be UML structured *DataType* not UML *Class*. **HasKey**(:*A* ($\text{OPE}_1 \ldots \text{OPE}_{mA}$) ($\text{DPE}_1 \ldots \text{DPE}_{nA}$)) **HasKey**(:*B* ($\text{OPE}_1 \ldots \text{OPE}_{mB}$) ($\text{DPE}_1 \ldots \text{DPE}_{nB}$)) **HasKey**(:*C* ($\text{OPE}_1 \ldots \text{OPE}_{mC}$) ($\text{DPE}_1 \ldots \text{DPE}_{nC}$)) **HasKey**(:*D* ($\text{OPE}_1 \ldots \text{OPE}_{mD}$) ($\text{DPE}_1 \ldots \text{DPE}_{nD}$))	Table 2 **VR1**
Transformation of UML binary Associations between two different Classes	
If the domain ontology contains any of below defined **AsymmetricObjectProperty** axioms, the defined UML Association is incorrect. **AsymmetricObjectProperty**(:*b*) **AsymmetricObjectProperty**(:*c*) **AsymmetricObjectProperty**(:*cR1*) **AsymmetricObjectProperty**(:*dR1*) **AsymmetricObjectProperty**(:*cR2*) **AsymmetricObjectProperty**(:*dR2*)	Table 6 **VR1**
If the domain ontology contains any of the below-defined **ObjectPropertyDomain** axioms where class expression is different than the given UML *Class*, the *Association* is defined in the ontology but between different *Classes*, than it is specified on the diagram. **ObjectPropertyDomain**(:*b* CE), where CE $\neq$:*C* **ObjectPropertyDomain**(:*c* CE), where CE $\neq$:*B* **ObjectPropertyDomain**(:*cR1* CE), where CE $\neq$:*D* **ObjectPropertyDomain**(:*dR1* CE), where CE $\neq$:*C* **ObjectPropertyDomain**(:*cR2* CE), where CE $\neq$:*D* **ObjectPropertyDomain**(:*dR2* CE), where CE $\neq$:*C*	Table 6 **VR2**
If the domain ontology contains any of below-defined **ObjectPropertyRange** axioms where the class expression is different than the given UML *Class*, the *Association* is defined in the ontology but between different *Classes*. **ObjectPropertyRange**(:*b* CE), where CE $\neq$:*B* **ObjectPropertyRange**(:*c* CE), where CE $\neq$:*C* **ObjectPropertyRange**(:*cR1* CE), where CE $\neq$:*C* **ObjectPropertyRange**(:*dR1* CE), where CE $\neq$:*D* **ObjectPropertyRange**(:*cR2* CE), where CE $\neq$:*C* **ObjectPropertyRange**(:*dR2* CE), where CE $\neq$:*D*	Table 6 **VR3**
Transformation of UML multiplicity of Association ends	

If the verification query returns a number greater than 0, it means that UML multiplicity is in contradiction with the domain ontology (?vioInd lists individuals that cause the violation).
SELECT ?**vioInd** (**count** (?**range**) **as** ?n)
WHERE { ?**vioInd** :*b* ?**range** } **GROUP BY** ?**vioInd**
HAVING (?n > 5)
SELECT ?**vioInd** (**count** (?**range**) **as** ?n)
WHERE { ?**vioInd** :*c* ?**range** } **GROUP BY** ?**vioInd**
HAVING (?n > 12)
SELECT ?**vioInd** (**count** (?**range**) **as** ?n)
WHERE { ?**vioInd** :*dR1* ?**range** } **GROUP BY** ?**vioInd**
HAVING (?n > 1)
	Table 9 **VR1**

SELECT ?**vioInd** (**count** (?**range**) **as** ?n)
WHERE { ?**vioInd** :*cR1* ?**range** } **GROUP BY** ?**vioInd**
HAVING (?n > 1)
SELECT ?**vioInd** (**count** (?**range**) **as** ?n)
WHERE { ?**vioInd** :*dR2* ?**range** } **GROUP BY** ?**vioInd**
HAVING (?n > 1)
SELECT ?**vioInd** (**count** (?**range**) **as** ?n)
WHERE { ?**vioInd** :*cR2* ?**range** } **GROUP BY** ?**vioInd**
HAVING (?n > 1)	

If the domain ontology contains **FunctionalObjectProperty** axiom specified for the association end which multiplicity is different from 1, the multiplicity is incorrect. **FunctionalObjectProperty**(:*b*) **FunctionalObjectProperty**(:*c*)	Table 9 **VR2**
If the domain ontology contains **SubClassOf** axiom, which specifies class expression with different multiplicity of the association ends than is derived from the UML class diagram, the multiplicity is incorrect. **SubClassOf**(:*C* CE), where CE ≠ **ObjectExactCardinality**(5 :*b* :*B*) **SubClassOf**(:*B* CE), where CE ≠ **ObjectUnionOf**(**ObjectExactCardinality**(7 :*c* :*C*) **ObjectIntersectionOf**(**ObjectMinCardinality**(10 :*c* :*C*) **ObjectMaxCardinality**(12 :*c* :*C*))) **SubClassOf**(:*C* CE), where CE ≠ **ObjectExactCardinality**(1 :*dR1* :*D*) **SubClassOf**(:*D* CE), where CE ≠ **ObjectExactCardinality**(1 :*cR1* :*C*) **SubClassOf**(:*C* CE), where CE ≠ **ObjectExactCardinality**(1 :*dR2* :*D*) **SubClassOf**(:*D* CE), where CE ≠ **ObjectExactCardinality**(1 :*cR2* :*C*)	Table 9 **VR3**
Transformation of UML Generalization between Classes	
If the domain ontology contains the defined **SubClassOf** axiom specified for *Classes*, which take part in the generalization relationship, the generalization relationship should be inverted on the diagram. **SubClassOf**(:*A* :*B*)	Table 12 **VR1**
Transformation of UML Generalization between Associations	
If the domain ontology contains the defined **SubObjectPropertyOf** axioms specified for *Association*, which take part in the generalization relationship, the generalization relationship should be inverted on the diagram. **SubObjectPropertyOf**(:*cR1* :*cR2*) **SubObjectPropertyOf**(:*dR1* :*dR2*)	Table 13 **VR1**

Example 2

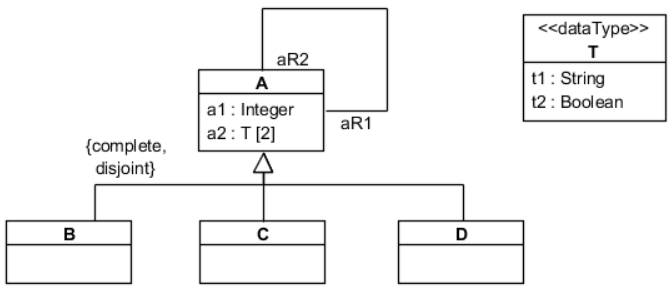


Figure 2. Example 2 of UML class diagram (see Tables 24, 25)

Table 24. Transformational part of UML class diagram from Example 2

Set of transformation axioms	Transformation rules
Transformation of UML Classes	
Declaration(**Class**(:*A*))	Table 2 **TR1**
Declaration(**Class**(:*B*))	
Declaration(**Class**(:*C*))	
Declaration(**Class**(:*D*))	
Transformation of UML attributes	
Declaration(**DataProperty**(:*a1*))	Table 4 **TR1**
Declaration(**ObjectProperty**(:*a2*))	
DataPropertyDomain(:*a1* :*A*)	Table 4 **TR2**
ObjectPropertyDomain(:*a2* :*A*)	
DataPropertyRange(:*a1* **xsd**:**integer**)	Table 4 **TR3**
ObjectPropertyRange(:*a2* :*T*)	Table 18 **TR2**
Transformation of UML multiplicity of attributes	
SubClassOf(:*A* **ObjectExactCardinality**(2 :*a2* :*T*))	Table 5 **TR1**
Transformation of UML binary Association from the Class to itself	
Declaration(**ObjectProperty**(:*aR1*))	
Declaration(**ObjectProperty**(:*aR2*))	Table 7 **TR1**
ObjectPropertyDomain(:*aR1* :*A*)	
ObjectPropertyDomain(:*aR2* :*A*)	Table 7 **TR2**
ObjectPropertyRange(:*aR1* :*A*)	
ObjectPropertyRange(:*aR2* :*A*)	Table 7 **TR3**
InverseObjectProperties(:*aR1* :*aR2*)	Table 7 **TR4**
AsymmetricObjectProperty(:*aR1*)	Table 7 **TR5**
AsymmetricObjectProperty(:*aR2*)	
Transformation of UML multiplicity of Association ends	
SubClassOf(:*A* **ObjectExactCardinality**(1 :*aR1* :*A*))	Table 9 **TR1**
SubClassOf(:*A* **ObjectExactCardinality**(1 :*aR2* :*A*))	
FunctionalObjectProperty(:*aR1*)	Table 9 **TR2**
FunctionalObjectProperty(:*aR2*)	
Transformation of UML Generalization between Classes	
SubClassOf(:*B* :*A*)	Table 12 **TR1**
SubClassOf(:*C* :*A*)	
SubClassOf(:*D* :*A*)	
Transformation of UML GeneralizationSet with {complete, disjoint} constraints	
DisjointUnion(:*A* :*B* :*C* :*D*)	Table 15 **TR1**
Transformation of UML structured DataType	
Declaration(**Class**(:*T*))	Table 19 **TR1**
Declaration(**DataProperty**(:*t1*))	Table 19 **TR2**
Declaration(**DataProperty**(:*t2*))	
DataPropertyDomain(:*t1* :*T*)	Table 19 **TR3**
DataPropertyDomain(:*t2* :*T*)	
DataPropertyRange(:*t1* **xsd**:**string**)	Table 19 **TR4**
DataPropertyRange(:*t2* **xsd**:boolean)	Table 18 **TR1**
	Table 18 **TR3**
HasKey(:*T* ( ) (:*t1* :*t2*))	Table 19 **TR5**

Table 25. Verificational part of UML class diagram from Example 2

Verificational part of UML class diagram	Verification rules
Transformation of UML Classes	
HasKey(:*A* (OPE$_1$…OPE$_{mA}$) (DPE$_1$…DPE$_{nA}$))	Table 2 **VR1**
HasKey(:*B* (OPE$_1$…OPE$_{mB}$) (DPE$_1$…DPE$_{nB}$))	
HasKey(:*C* (OPE$_1$…OPE$_{mC}$) (DPE$_1$…DPE$_{nC}$))	
HasKey(:*D* (OPE$_1$…OPE$_{mD}$) (DPE$_1$…DPE$_{nD}$))	
Transformation of UML attributes	
DataPropertyDomain(:*a1* CE), where CE $\neq$ A	Table 4 **VR1**
ObjectPropertyDomain(:*a2* CE), where CE $\neq$ A	
DataPropertyRange(:*a1* DR), where	Table 4 **VR2**
DR $\neq$ **xsd:integer** **ObjectPropertyRange**(:*a2* CE), where	Table 18 **TR2**
CE $\neq$:*T*	
Transformation of UML multiplicity of attributes	
SELECT ?**vioInd** (**count** (?**range**) **as** ?n)	Table 5 **VR1**
WHERE { ?**vioInd** :*a2* ?**range** } **GROUP BY** ?**vioInd**	
HAVING (?n > 2)	
SubClassOf(:*A* CE), where	Table 5 **VR2**
CE $\neq$ **ObjectExactCardinality**(2 :*a2* :*T*)	
Transformation of UML binary Association from the Class to itself	
ObjectPropertyDomain(:*aR1* CE), where CE $\neq$:*A*	Table 7 **VR1**
ObjectPropertyDomain(:*aR2* CE), where CE $\neq$:*A*	
ObjectPropertyRange(:*aR1* CE), where CE $\neq$:*A*	Table 7 **VR2**
ObjectPropertyRange(:*aR2* CE), where CE $\neq$:*A*	
Transformation of UML multiplicity of Association ends	
SELECT ?**vioInd** (**count** (?**range**) **as** ?n)	Table 9 **VR1**
WHERE { ?**vioInd** :*aR1* ?**range** } **GROUP BY** ?**vioInd**	
HAVING (?n > 1)	
SELECT ?**vioInd** (**count** (?**range**) **as** ?n)	
WHERE { ?**vioInd** :*aR2* ?**range** } **GROUP BY** ?**vioInd**	
HAVING (?n > 1)	
SubClassOf(:*A* CE), where CE $\neq$ **ObjectExactCardinality**(1 :*aR1* :*A*)	Table 9 **VR3**
SubClassOf(:*A* CE), where CE $\neq$ **ObjectExactCardinality**(1 :*aR2* :*A*)	
Transformation of UML Generalization between Classes	
SubClassOf(:*A* :*B*)	Table 12 **VR1**
SubClassOf(:*A* :*C*)	
SubClassOf(:*A* :*D*)	
Transformation of UML GeneralizationSet with {complete, disjoint} constraints	
SubClassOf(:*B* :*C*)	Table 15 **VR1**
SubClassOf(:*C* :*B*)	
SubClassOf(:*C* :*D*)	
SubClassOf(:*D* :*C*)	
SubClassOf(:*B* :*D*)	
SubClassOf(:*D* :*B*)	

Transformation of UML structured DataType	
Check if the :T class is specified in the domain ontology as a subclass (**SubClassOf** axiom) of any class expression, which does not have **HasKey** axiom defined.	Table 19 **VR1**

Example 3

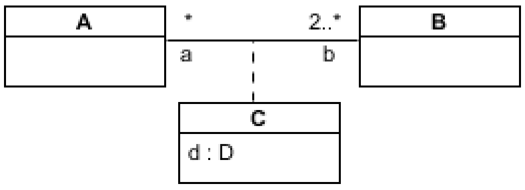


Figure 3. Example 3 of UML class diagram (see Tables 26 and 27)

Table 26. Transformational part of UML class diagram from Example 3

Set of transformation axioms	Transformation rules
Transformation of UML Classes	
Declaration(**Class**(:A))	Table 2 **TR1**
Declaration(**Class**(:B))	
Transformation of UML attributes	
Declaration(**ObjectProperty**(:d))	Table 4 **TR1**
ObjectPropertyDomain(:d :C)	Table 4 **TR2**
ObjectPropertyRange(:d :D)	Table 4 **TR3**
Transformation of UML binary Associations between two different Classes	
Declaration(**ObjectProperty**(:a))	Table 6 **TR1**
Declaration(**ObjectProperty**(:b))	
ObjectPropertyDomain(:a **ObjectUnionOf**(:B :C))	Table 6 **TR2**
ObjectPropertyDomain(:b **ObjectUnionOf**(:A :C))	Table 10 **TR1**
ObjectPropertyRange(:a :A)	Table 6 **TR3**
ObjectPropertyRange(:b :B)	
InverseObjectProperties(:a :b)	Table 6 **TR4**
Transformation of UML multiplicity of Association ends	
SubClassOf(:A **ObjectMinCardinality**(2 :b :B))	Table 9 **TR1**
Transformation of UML AssociationClass	
Declaration(**Class**(:C))	Table 10 **TR2**
Declaration(**ObjectProperty**(:c))	Table 10 **TR3**
ObjectPropertyDomain(:c **ObjectUnionOf**(:A :B))	Table 10 **TR4**
ObjectPropertyRange(:c :C)	Table 10 **TR5**

Table 27. Verificational part of UML class diagram from Example 3

Verificational part of UML class diagram	Verification rules
Transformation of UML Classes	
HasKey(:A (OPE$_1$ ...OPE$_m$) (DPE$_1$ ...DPE$_n$)) **HasKey**(:B (OPE$_1$ ...OPE$_m$) (DPE$_1$ ...DPE$_n$))	Table 2 **VR1**
Transformation of UML attributes	
ObjectPropertyDomain(:d CE), where CE$\neq$:C	Table 4 **VR1**
ObjectPropertyRange(:d CE), where CE$\neq$:D	Table 4 **VR2**
Transformation of UML binary Associations between two different Classes	
AsymmetricObjectProperty(:a) **AsymmetricObjectProperty**(:b)	Table 6 **VR1**
Transformation of UML multiplicity of Association ends	
FunctionalObjectProperty(:a) **FunctionalObjectProperty**(:b)	Table 9 **VR2**
SubClassOf(:A CE), where CE$\neq$**ObjectMinCardinality**(2 :b :B) **SubClassOf**(:B CE), where CE = any explicitly specified multiplicity	Table 9 **VR3**
Transformation of UML AssociationClass	
HasKey(:C (OPE$_1$ ...OPE$_m$) (DPE$_1$ ...DPE$_n$))	Table 10 **VR1**
ObjectPropertyDomain(:a CE), where CE$\neq$**ObjectUnionOf**(:B :C) **ObjectPropertyDomain**(:b CE), where CE$\neq$**ObjectUnionOf**(:A :C) **ObjectPropertyDomain**(:c CE), where CE$\neq$**ObjectUnionOf**(:A :B)	Table 10 **VR2**
ObjectPropertyRange(:c CE), where CE$\neq$:C	Table 10 **VR3**

8. Tool support for validation and automatic correction of UML class diagrams

The transformation and verification rules presented in Section 5 have been implemented in a tool. All the defined rules are proved to be fully implementable. As a result, the tool allows one to transform any UML class diagram built of different kinds of UML elements (listed in Section 5, and selected based on their importance from the perspective of pragmatics) to OWL 2 representation. In comparison to other available tools which allow transforming UML class diagrams to an OWL 2 representation, the range of the transformed constructions is wider as it benefits from the results of the conducted systematic literature review and its analysis, revision and extension.

Due to the fact that the tool is still under development, the following webpage has been created in which the tool with the installation instructions will be later accessible: https://sourceforge.net/projects/uml-class-diagrams-validation/

After the development and experiment phases are finished, the tool will be made available online.

The tool has been tested with a number of test cases aimed to determine whether the tool fully and correctly implemented the transformation and verification rules, as well as the validation method. At least one test case has been prepared for every normalization, transformation and verification rule. Additionally, a number of test cases have been prepared to cover popular assemblies of UML elements, e.g. an association from a class to itself, an association between two classes, two associations between two classes, two associations between three classes, etc. Each rule has been independently checked if it returns the expected result. In total, the number of test cases was as follows:
1. 80 test cases for ontology normalization rules,
2. 40 test cases for transformation rules,

3. 25 test cases for verification rules.

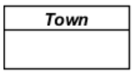
The tool passed all test cases.

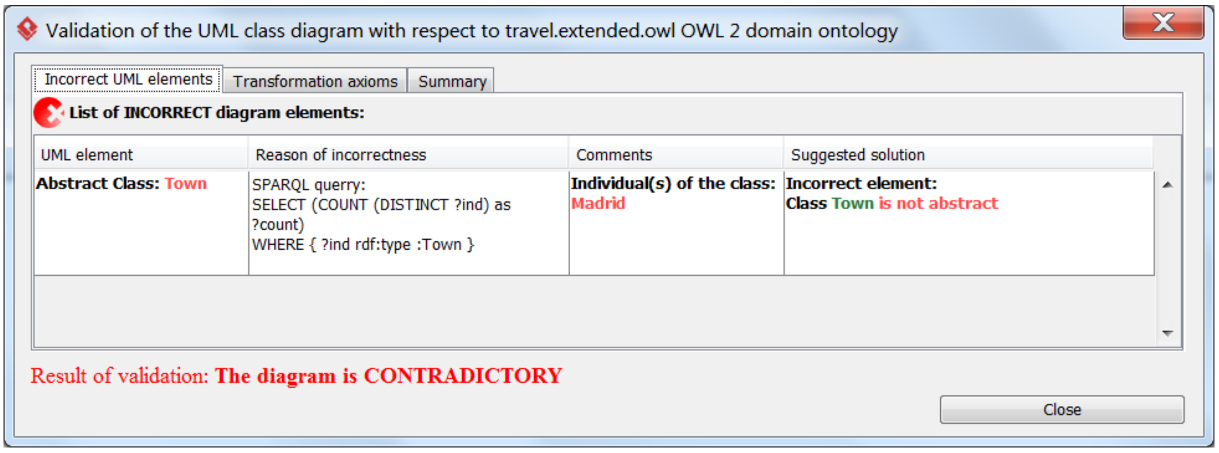
The implementation of the transformation and verification rules as well as the method of validation explained in [1], resulted in a functionality of the tool allowing for validation if a UML class diagram is compliant with the selected domain ontology. Furthermore, on the basis of the result of validation, the tool automatically generates ontology-based suggestions for diagram corrections. In [4], a few initial suggestions for diagram corrections have been presented. The initial list of suggestions has been revised and extended, and currently the tool automatically generates suggestions of two kinds:

1. What has to be corrected in the UML class diagram in order for the diagram not to be contradictory with the selected domain ontology (approximately 30 types of suggestions – one for violation of every verification rules plus one general suggestion listing incorrect UML elements if a transformation rule has caused the inconsistency in the domain ontology). This list of suggestions is reported by the tool for the modeller and strongly advised his or her attention (Example 4 and 5).
2. Based on the domain ontology of what might be additionally included in the class diagram (9 types of suggestions). Whether or not to consider this list of proposed suggestions is for the modeller to decide. Depending on the specific requirements, the suggestions may be incorporated in the diagram (Example 6).

Example 4

Table 28. Example of what has to be corrected in the diagram based on the ontology: abstract class verification

Suggestion: the class is not abstract	
Axiom(s) in the OWL domain ontology	**Declaration**(**Class**(:*Town*)) **ClassAssertion**(:*Town* :*Madrid*)
Element on the UML class diagram	
Result of validation:	



Validation of the UML class diagram with respect to travel.extended.owl OWL 2 domain ontology

Incorrect UML elements | Transformation axioms | Summary

List of INCORRECT diagram elements:

UML element	Reason of incorrectness	Comments	Suggested solution

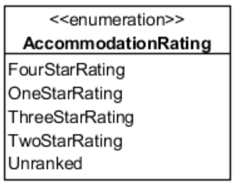
Abstract Class: Town	SPARQL querry:
	SELECT (COUNT (DISTINCT ?ind) as ?count)
	WHERE { ?ind rdf:type :Town }
		Individual(s) of the class: Madrid	**Incorrect element:**
			Class Town is not abstract

Result of validation: **The diagram is CONTRADICTORY**

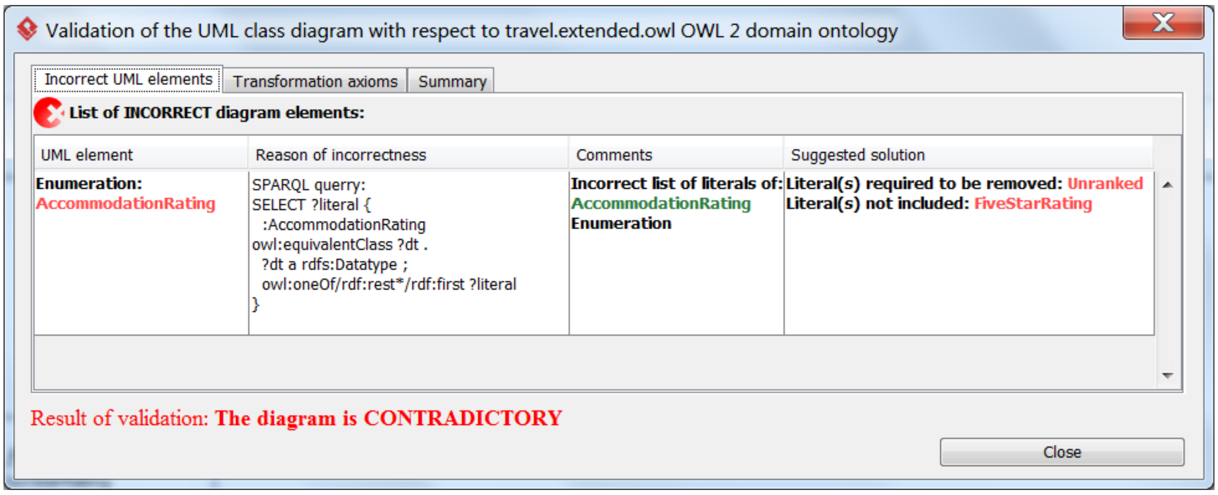
Close

Example 5

Table 29. Example of what has to be corrected in the diagram based on the ontology: enumeration verification

Suggestion: the enumeration is incorrectly defined	
Axiom(s) in the OWL domain ontology	**DatatypeDefinition**(:*AccommodationRating* **DataOneOf**("OneStarRating" "TwoStarRating" "ThreeStarRating" "FourStarRating" "FiveStarRating"))
Element on the UML class diagram	 <<enumeration>> AccommodationRating FourStarRating OneStarRating ThreeStarRating TwoStarRating Unranked

Result of validation:



Validation of the UML class diagram with respect to travel.extended.owl OWL 2 domain ontology

Incorrect UML elements | Transformation axioms | Summary

List of INCORRECT diagram elements:

UML element	Reason of incorrectness	Comments	Suggested solution

Enumeration: AccommodationRating	SPARQL querry:
	SELECT ?literal {
	:AccommodationRating
	owl:equivalentClass ?dt .
	?dt a rdfs:Datatype ;
	owl:oneOf/rdf:rest*/rdf:first ?literal
	}
		Incorrect list of literals of: AccommodationRating **Enumeration**	**Literal(s) required to be removed:** Unranked
			Literal(s) not included: FiveStarRating

Result of validation: **The diagram is CONTRADICTORY**

Close

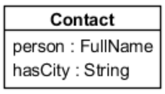
Example 6

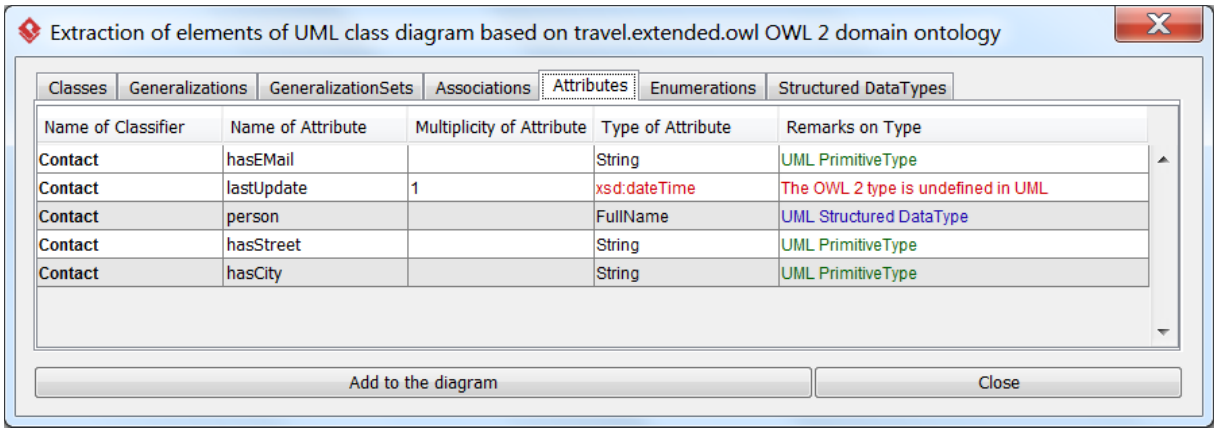
Table 30. Example of what may be incorporated in the diagram based on the ontology: attribute verification

Suggestion: Insert missing attribute, missing type of attribute or missing multiplicity of attribute	
Axiom(s) in the OWL domain ontology	**Declaration**(**Class**(:*Contact*)) **ObjectPropertyDomain**(:*person* :*Contact*) **ObjectPropertyRange**(:*person* :*FullName*) **Declaration**(**Class**(:*FullName*)) **DataPropertyDomain**(:*firstName* :*FullName*) **DataPropertyRange**(:*firstName* **xsd:string**) **DataPropertyDomain**(:*secondName* :*FullName*) **DataPropertyRange**(:*secondName* **xsd:string**) **HasKey**(:*FullName* ( ) (:*firstName* :*secondName*)) **Declaration**(**DataProperty**(:*hasEMail*)) **DataPropertyDomain**(:*hasEMail* :*Contact*) **DataPropertyRange**(:*hasEMail* **xsd:string**)

Declaration(**DataProperty**(:*hasStreet*))
DataPropertyDomain(:*hasStreet* :*Contact*)
DataPropertyRange(:*hasStreet* **xsd:string**)
Declaration(**DataProperty**(:*hasCity*))
DataPropertyDomain(:*hasCity* :*Contact*)
DataPropertyRange(:*hasCity* **xsd:string**)
Declaration(**DataProperty**(:*lastUpdate*))
DataPropertyDomain(:*lastUpdate* :*Contact*)
DataPropertyRange(:*lastUpdate* **xsd**:dateTime)
SubClassOf(:*Contact* **DataExactCardinality**(1 :*lastUpdate*))

Element on the UML class diagram





Name of Classifier	Name of Attribute	Multiplicity of Attribute	Type of Attribute	Remarks on Type
Contact	hasEMail		String	UML PrimitiveType
Contact	lastUpdate	1	xsd:dateTime	The OWL 2 type is undefined in UML
Contact	person		FullName	UML Structured DataType
Contact	hasStreet		String	UML PrimitiveType
Contact	hasCity		String	UML PrimitiveType

Legend:
white rows – suggestions of UML elements which might be included in the class diagram
grey rows – UML elements already included in the class diagram

9. Conclusions

The paper presents rules for transforming UML class diagrams to their OWL 2 representations. All the static elements of UML class diagrams commonly used in business or conceptual modelling have been considered. The vast majority of the elements can be fully transformed to OWL 2 constructs. The presented transformation rules result from an in-depth analysis and extension of the state-of-the-art transformation rules identified through a systematic literature review. In total, 41 transformation rules have been described (not counting our complementation to the rules of disjointness presented in Section 6). 25 transformation rules have been directly extracted from the literature, 8 rules originate in the literature but have been extended by us in order to reflect the semantics of UML elements in OWL more precisely, and 8 transformation rules are our new propositions.

In addition to the transformation rules, we have defined all the presented verification rules (26 in total). The verification rules are aimed at checking the compliance of the OWL representation of UML class diagram with the given OWL domain ontology. The described transformation and verification rules are crucial in the method of semantic validation of UML class diagrams [1]. The approach validates automatically if a selected class diagram is compliant with the selected OWL 2 domain ontology.

The developed method and the tool are a pragmatic attempt of bringing together the differences in the philosophy of UML and OWL 2 languages. In order to make the process automatic, the tool has been supplemented with all the transformation and verification rules, and has been tested with a wide range of test cases. The inclusions to the tool are the result of pragmatic thinking. For example, the implementation allowed observing that it is worth extending the tool so that it automatically generates ontology-based suggestions for diagram corrections.

The tool already offers a range of new possibilities for practical application of domain ontologies. However, as a consequence, the proposed approach creates a need for greater involvement of domain ontologies in modeling.

The research background of our considerations can be supported by other publications, e.g. [35–37]. The potential of reusing domain ontologies for the purpose of validation is promising and may help the modelers through automation. The choice of OWL is justified by the growing number of the already created ontologies in this language. For future work, the development of the tool is planned to be finished soon. The next step of work is preparation of experiment aimed at validation of the tool and the method in practice. The experiment is aimed to state the practicality of our proposal.

References

[1] M. Sadowska and Z. Huzar, "Semantic validation of UML class diagrams with the use of domain ontologies expressed in OWL 2," in *Software Engineering: Challenges and Solutions*. Springer International Publishing, 2016, pp. 47–59.

[2] *Unified Modeling Language, Version 2.5*, OMG, 2015. [Online]. http://www.omg.org/spec/UML/2.5

[3] *OWL 2 Web Ontology Language Document Overview (Second Edition)*, W3C, 2012. [Online]. https://www.w3.org/TR/owl2-overview/

[4] M. Sadowska, "A prototype tool for semantic validation of UML class diagrams with the use of domain ontologies expressed in OWL 2," in *Towards a Synergistic Combination of Research and Practice in Software Engineering*. Springer International Publishing, 2017, pp. 49–62.

[5] M. Sadowska and Z. Huzar, "The method of normalizing OWL 2 DL ontologies," *Global Journal of Computer Science and Technology*, Vol. 18, No. 2, 2018, pp. 1–13.

[6] A. Korthaus, "Using UML for business object based systems modeling," in *The Unified Modeling Language*. Physica-Verlag HD, 1998, pp. 220–237.

[7] H.E. Eriksson and M. Penker, *Business Modeling With UML: Business Patterns at Work*. New York, USA: John Wiley & Sons, Inc., 2000.

[8] E.D. Nitto, L. Lavazza, M. Schiavoni, E. Tracanella, and M. Trombetta, "Deriving executable process descriptions from UML," in *Proceedings of the 24th International Conference on Software Engineering*, ICSE '02. New York, NY, USA: ACM, 2002, pp. 155–165.

[9] C. Fu, D. Yang, X. Zhang, and H. Hu, "An approach to translating OCL invariants into OWL 2 DL axioms for checking inconsistency," *Automated Software Engineering*, Vol. 24, No. 2, 2017, pp. 295–339.

[10] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University & University of Durham, EBSE Technical Report EBSE 2007-01, 2007.

[11] X. Zhou, Y. Jin, H. Zhang, S. Li, and X. Huang, "A map of threats to validity of systematic literature reviews in software engineering," in *23rd Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2016, pp. 153–160.

[12] Z. Xu, Y. Ni, W. He, L. Lin, and Q. Yan, "Automatic extraction of OWL ontologies from UML class diagrams: a semantics-preserving approach," *World Wide Web*, Vol. 15, No. 5, 2012, pp. 517–545.

[13] Z. Xu, Y. Ni, L. Lin, and H. Gu, "A semantics-preserving approach for extracting OWL ontologies from UML class diagrams," in *Database Theory and Application*, Communications in Computer and Information Science. Berlin, Heidelberg: Springer, 2009, pp. 122–136.

[14] M. Mehrolhassani and A. Elçi, "Developing ontology based applications of semantic web using UML to OWL conversion," in *The Open Knowlege Society. A Computer Science and Information Systems Manifesto*, Communications in Computer and Information Science. Berlin, Heidelberg: Springer, 2008, pp. 566–577.

[15] O. El Hajjamy, K. Alaoui, L. Alaoui, and M. Bahaj, "Mapping UML to OWL2 ontology," *Journal of Theoretical and Applied Information Technology*, Vol. 90, No. 1, 2016, pp. 126–143.

[16] C. Zhang, Z.R. Peng, T. Zhao, and W. Li, "Transformation of transportation data models from Unified Modeling Language to Web Ontology Language," *Transportation Research Record: Journal of the Transportation Research Board*, Vol. 2064, No. 1, 2008, pp. 81–89.

[17] J. Zedlitz, J. Jörke, and N. Luttenberger, "From UML to OWL 2," in *Knowledge Technology*,

Communications in Computer and Information Science. Berlin, Heidelberg: Springer, 2012, pp. 154–163.

[18] A.H. Khan and I. Porres, "Consistency of UML class, object and statechart diagrams using ontology reasoners," *Journal of Visual Languages & Computing*, Vol. 26, 2015, pp. 42–65.

[19] A.H. Khan, I. Rauf, and I. Porres, "Consistency of UML class and statechart diagrams with state invariants," in *Proceedings of the 1st International Conference on Model-Driven Engineering and Software Development*, S. Hammoudi, L.F. Pires, J. Filipe, and R.C. das Neves, Eds., Vol. 1. SciTePress Digital Library, 2013, p. 1–11.

[20] J. Zedlitz and N. Luttenberger, "Transforming between UML conceptual models and OWL 2 ontologies," in *Terra Cognita 2012 Workshop*, Vol. 6, 2012, p. 15.

[21] W. Xu, A. Dilo, S. Zlatanova, and P. van Oosterom, "Modelling emergency response processes: Comparative study on OWL and UML," in *Proceedings of the Joint ISCRAM-CHINA and GI4DM Conference*, Harbin, China, 2008, pp. 493–504.

[22] N. Gherabi and M. Bahaj, "A new method for mapping UML class into OWL ontology," *International Journal of Computer Applications Special Issue on Software Engineering, Databases and Expert Systems*, Vol. SEDEXS, No. 1, 2012, pp. 5–9. [Online]. https://research.ijcaonline.org/sedex/number1/sedex1002.pdf

[23] H.S. Na, O.H. Choi, and J.E. Lim, "A method for building domain ontologies based on the transformation of UML models," in *Fourth International Conference on Software Engineering Research, Management and Applications (SERA'06)*, D.K. Baik, D. Primeaux, N. Ishii, and R. Lee, Eds., IEEE, 2006, pp. 332–338.

[24] M. Bahaj and J. Bakkas, "Automatic conversion method of class diagrams to ontologies maintaining their semantic features," *International Journal of Soft Computing and Engineering*, Vol. 2, No. 6, 2013, pp. 65–69.

[25] A. Belghiat and M. Bourahla, "Transformation of UML models towards OWL ontologies," in *2012 6th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT)*, 2012, pp. 840–846.

[26] S. Höglund, A.H. Khan, Y. Liu, and I. Porres, "Representing and validating metamodels using OWL 2 and SWRL," in *Proceedings of the 9th Joint Conference on Knowledge-Based Software Engineering*, 2010.

[27] K. Kiko and C. Atkinson, "A detailed comparison of UML and OWL," University of Mannheim, Fakultät für Mathematik und Informatik, Lehrstuhl für Softwaretechnik, Tech. Rep. TR-2008-004, 2008.

[28] J. Zedlitz and N. Luttenberger, "Data types in UML and OWL-2," in *The Seventh International Conference on Advances in Semantic Processing*, 2013, pp. 32–35.

[29] J. Zedlitz and N. Luttenberger, "Conceptual modelling in UML and OWL-2," *International Journal on Advances in Software*, Vol. 7, No. 1 & 2, 2014, pp. 182–196.

[30] *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition)*, W3C, 2012. [Online]. https://www.w3.org/TR/owl2-syntax/

[31] *OWL 2 Web Ontology Language New Features and Rationale (Second Edition)*, W3C, 2012. [Online]. https://www.w3.org/TR/owl2-new-features/

[32] N. Noy and A. Rector, *Defining N-ary Relations on the Semantic Web*, W3C, 2006. [Online]. https://www.w3.org/TR/swbp-n-aryRelations/

[33] *W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes*, W3C, 2012. [Online]. https://www.w3.org/TR/xmlschema11-2/

[34] *OWL 2 Web Ontology Language Primer (Second Edition)*, W3C, 2012. [Online]. https://www.w3.org/TR/owl2-primer/

[35] I. Dubielewicz, B. Hnatkowska, Z. Huzar, and L. Tuzinkiewicz, "Domain modeling in the context of ontology," *Foundations of Computing and Decision Sciences*, Vol. 40, No. 1, 2015, pp. 3–15. [Online]. https://content.sciendo.com/view/journals/fcds/40/1/article-p3.xml

[36] B. Hnatkowska, Z. Huzar, L. Tuzinkiewicz, and I. Dubielewicz, "A new ontology-based approach for construction of domain model," in *Intelligent Information and Database Systems*, N.T. Nguyen, S. Tojo, L.M. Nguyen, and B. Trawiński, Eds., Cham: Springer International Publishing, 2017, pp. 75–85.

[37] I. Dubielewicz, B. Hnatkowska, Z. Huzar, and L. Tuzinkiewicz, "Domain modeling based on requirements specification and ontology," in *Software Engineering: Challenges and Solutions*, L. Madeyski, M. Śmiałek, B. Hnatkowska, and Z. Huzar, Eds., Cham: Springer International Publishing, 2017, pp. 31–45.

e-Informatica Software Engineering Journal, Volume 13, Issue 1, 2019, pages: 105–139, DOI 10.5277/e-Inf190104

A Three Dimensional Empirical Study of Logging Questions from Six Popular Q & A Websites

Harshit Gujral*, Abhinav Sharma*, Sangeeta Lal*, Lov Kumar**
*Jaypee Institute of Information Technology, Noida, Uttar-Pradesh, India
**BITS-pilani Hydrabad Campus, Hydrabad, India
harshitgujral12@gmail.com, sharma1997abhinav@gmail.com, sangeeta@jiit.ac.in, lovkumar505@gmail.com

Abstract

Background: Q & A websites such as Stack Overflow, Server Fault, provide an open platform for users to ask questions and to get help from experts present worldwide. These websites not only help users by answering their questions but also act as a knowledge base. These data present on these websites can be mined to extract valuable information that can benefit the software practitioners. Software engineering research community has already understood the potential benefits of mining data from Q & A websites and several research studies have already been conducted in this area.
Aim: The aim of the study presented in this paper is to perform an empirical analysis of logging questions from six popular Q & A websites.
Method: We perform statistical, programming language and content analysis of logging questions. Our analysis helped us to gain insight about the logging discussion happening in six different domains of the Stack Exchange websites.
Results: Our analysis provides insight about the logging issues of software practitioners: logging questions are pervasive in all the Q & A websites, the mean time to get accepted answer for logging questions on SU and SF websites are much higher as compared to other websites, a large number of logging question invite a great amount of discussion in the SoftwareEngineering Q & A website, most of the logging issues occur in C++ and Java, the trend for number of logging questions is increasing for Java, Python, and JavaScript, whereas, it is decreasing or constant for C, C++, C#, for the Server Fault and Superuser website 'C' is the dominant programming language.

Keywords: classification, debugging, ensemble, logging, machine learning, source code analysis, tracing

1. Introduction

Logging is an important programming practice that is performed by inserting log statements in the source code. These log statements are used to record important runtime information about the program execution. Software developers can use this runtime information at the time of debugging. In addition to debugging, logging is important in several other software development activities such as anomaly detection [1], performance problem diagnosis [2]. For example, Fu et al. [1] use log messages timings to differentiate normal and anomalous executions. Nagaraj et al. [2] purpose a system that compares the state of normal execution sequence (normal performance) and bad execution sequences (bad performance) and identify the states that are different between the two execution sequences. Logging is an important activity for software development, however software developers often face challenges In logging due to changing nature of source code as well as logging libraries. For example, software developers face difficulty

Submitted: 4 August 2018; Revised: 28 February 2019; Accepted: 28 February 2019; Available online: 31 May 2019

in identifying code constructs that needs to be logged [3, 4], log level that needs to assigned to log statements [5] or issues in migrating log libraries [6]. Hence, recently several techniques have been proposed by the software engineering research community to help software developers in source code logging [3–5, 7].

The techniques proposed in the literature for helping software developers in logging are useful, but, at present there is little understanding about the major logging concerns of the different software practitioners like software developers, system administrators, database administrators, etc. A detailed study of the most frequent logging concerns of the software practitioners can be beneficial in further improving the existing logging techniques or tools. Information present on the technical Q & A websites can be a great resource for identifying the logging concerns of the software practitioners. Table 1 shows 6 logging questions from six popular Stack Exchange Q & A websites, i.e. Stack Overflow (SO) [8], Server Fault (SF) [9], SuperUser (SU) [10], Database Administrators (DB) [11], Android Enthusiast (AE) [12], and Software Engineering (SE) [13]. Each question in Table 1 received thousands of views from the software development community. For example, question 1 received 192,320 views. This indicates the impact and reach of Q & A websites in software development community. In the question 1, the user has asked a questions on SF website which is related to 'Enabling MySQL logging'. It shows that users face issue in enabling MySQL logging. In question 6, the user has asked about 'best practices of logging user actions in production'. In this question, user wants more information about logging practices of user action. We believe that a detailed characterization study of the logging questions asked on these websites can provide a valuable insights about the logging needs of software development community.

The software engineering research community has already recognized the potential of the Q & A websites in various applications [14, 15]. For example, Pinto et al. [14] analyze the SO questions to find application-level energy consumption related issues. Mario et al. [15] analyze SO questions to find mobile development related issues. Barua et al. [16] analyze questions on the SO website to find software development related trends. All these studies analyze important aspects of software development. However, at present there is no research study that analyzes the data from the Q & A website for identifying source code logging issues. In this paper, we take the first step towards analyzing the logging concerns of software developers from popular Q & A websites.

The overall goal of our research is to improve the understanding about the logging issues that software practitioners face the most. In particular, we aim at systematically analyzing the questions from Q & A websites. We hypothesize that Q & A websites represents an important knowledge base and can be beneficial in identifying the source code logging concerns of the software developers. The findings of this paper can be beneficial to software practitioners in many ways. Product manager can use this study to perform market analysis to find logging tools that are gaining popularity. Software practitioners can use this study to find logging tools/libraries that are commonly used by other software practitioners. These findings can be used by the Stack Exchange team for site moderation/archiving purpose. Additionally, software engineering research community can use the results presented in this paper to further improve the current logging prediction or improvement studies.

In this work, we perform a three dimensional, large scale and an in-depth empirical study of logging questions asked on six popular community based Q & A websites from the Stack Exchange network. We analyze more than 82 K questions from six popular programming Q & A websites with respect to three different research dimensions, i.e. *statistical analysis*, *programming language analysis*, and *content analysis* and answer a total of 7 research questions. The results of our empirical analysis show several interesting insights such as logging questions are pervasive in all the programming websites. It shows that

Table 1. Example of logging related questions from various websites: WN: Website Name

S.No.	WN	Question Id	Title	View Count	Tags
1	SF	71071	How to enable MySQL logging?	192,320	MySQL, **logging**
2	SO	56628	How do you clear the SQL Server transaction log?	965,799	sql-server, **transaction-log**
3	SU	176165	Where Linux places the messages of boot?	139,650	Linux, boot, centos, **logging**
4	AE	14430	How can I view and examine the Android log?	355,050	**logging**
5	DB	4043	Can I see Historical Queries run on a SQL Server database?	173,358	sql-server, SQL, **logs**
6	SE	168059	Best practices for logging user actions in production	49,435	C#, asp.net, **logging**

nearly 1.06–11.6% of all the logging questions invite a great amount of discussion. The work presented in this paper is a significant extension of our previously accepted work *Empirical Analysis of the Logging Questions on the Stack Overflow Website* at *Conference on Software Engineering & Data Sciences (CoSEDS-2018).*

The remainder of the paper is organized as follows. In Section 2, we describe the closely related studies in context to the work presented in this paper and the novel research contribution made by this work. In Section 3, we describe the various research dimensions and their respective research questions, research method that we followed, the experimental dataset, and the results of the empirical study. In Section 4, we give various threats to validity related to the finding of this paper. In Section 5 we conclude the paper and provide details about future directions and finally, in Section 6, we give acknowledgment.

2. Related work

In this section, we review the closely related work to our research and list down our specific research contributions. We divide related work into multiple lines of research, i.e. 1) Empirical analysis of logging statement, 2) Logging prediction studies, and 3) Empirical analysis of Q & A websites.

2.1. Empirical analysis of logging statement

Logging is a cross-cutting software development concern and has attracted attention of many researchers. Logging statements present in the source code have been analyzed with respect to several dimensions, such as **type of changes in log statements** ([17]), **reasons of migrating from one logging library to other** ([6]), **source code constructs that are logged more frequently as compared to others** ([18, 19]), **relationship between code quality and logging statements** [20], **uses of different log levels in source code** ([5]). These analysis provide important information to software developers. For example, Yuan et al. [21] analyze four open source projects written in C\C++. They identify type of changes to logging statements where software developers spend most of their time. Shang et al. [20] analyze modifications done to logging statements for Java projects. They report four major reasons for logging modification, i.e. debugging, feature change, inaccurate logging level, and redundant logging. Chen et al. [17] replicate the study performed by Yuan et al. [21] for Java projects and report several differences in the results. For example, in Java projects deleting and moving log printing code accounts to 26% and 10% to all logging modifications whereas in C\C++ it accounts to only

2%. Kabinna et al. [6] identify reasons for logging library migration on Java software projects. They report two major reasons, i.e. flexibility and performance improvement, for logging library migration. Li et al. [5] analyze log-levels of various open-source Java projects and report several interesting findings. They report that no single log level dominates. They also report that different projects show varying distribution of log levels. In another study, Li et al. [22] analyze four Java software projects and identify 20 reasons for logging change in source code. They categorize these 20 reasons into four categories: *changing context code*, *improving logging*, *dependency-driven changes*, and *fixing logging issues*. Yuan et al. [23] analyze 250 randomly sampled bug reports from five large C\C++ projects and report the most frequently occurring error patterns that need to be logged. Fu et al. [18] work on analyzing logged and non-logged code constructs. They analyze log statements and their logged code snippets from two closed-source systems at Microsoft (written in C#). They categorize the log statements in five categories: *assertion-check, return-value-check, exception, logic-branch* and *observing-point* logging. They further perform a detailed study of 70 non-logged catch-blocks and find reasons of not logging. Lal et al. [19] analyze logged and non-logged catch-blocks. They report several distinguishing characteristics between logged and non-logged catch-blocks. For example, try-blocks associated with logged catch-blocks have much higher complexity (measures using SLOC, number of operators and number of method calls) as compared to that of non-logged catch-blocks.

All of the above studies analyze logging statements present in the source code. In contrast to these studies, in this work, we analyze logging questions from six Stack Exchange sites to get insights about the logging issues that software developers face most frequently.

2.2. Logging prediction studies

Logging is crucial for software development and hence, in past researchers spent a great amount of effort for providing software developers with tools and techniques that can help them in source code Logging. For example, Fu et al. [18] and Zhu et al. [24] propose a tool *LogAdvsior* to help software developers in logging prediction for exception types and return value check code snippets for C# projects. Lal et at. [3, 4] propose *LogOpt* and *LogOptPlus*, machine learning models for catch-blocks and if-blocks logging prediction for Java projects. Li et al. [5] propose model for log level prediction. Kabinna et al. [25] propose a model for log statement stability prediction for Java projects. Lal et al. [7] propose a method *LogIm* for predicting logging statement for if and catch-blocks for imbalanced dataset. In another study, Lal et al. [26] use ensemble of classifiers for doing cross-project logging prediction.

The work presented in this paper, is complementary to above studies. We work on identifying the most frequent logging issues of the software developers. Hence, the findings of this work can be beneficial in further improving these logging prediction models. For example, researchers can select in which language software practitioners face most of the logging issues and can provide logging tools for the same. Researchers can identify which are the most frequent libraries in which software practitioners facing the issues and hence, can provide solutions to apply logging prediction for these logging libraries.

2.3. Empirical analysis of Q & A websites

The Stack Exchange is network of popular Q & A sites and is actually a knowledge base, several research studies have already been conducted using the data from Stack Exchange websites. Pinto et al. present an empirical study on analyzing 300 Stack Overflow questions and 550 Stack Overflow answers on problems related to application-level energy consumption [14]. They study distinctive characteristics, most common problems, main causes and solutions recommended on software energy consumption [14]. Mario et al. apply topic modelling to discover hot-topics on mobile development by mining questions and answers on Stack Overflow [15]. Their findings reveal that most of the questions are on compatibility issues, crash reports and database connection [15]. Beyer et al. conduct a manual categorization of Android app development related issues on Stack Overflow [27]. They investigate 450 Android related posts and conclude that developers mainly have issues related to usage of API components such as User Interface and Core Elements [27].

Yang et al. study security related questions on Stack Overflow and cluster security related questions (such as cryptography and mobile security) based on their text [28]. They discover that security related questions belong to five main categories, i.e. web security, mobile security, cryptography, software security, and system security [28]. Malik et al. manually analyse 1000 posts on Android energy consumption [29]. Their study reveals that most of the questions are related to improper implementation, sensor and radio utilization [29]. Nagy et al. present a study in mining Stack Overflow for discovering error-prone patterns in SQL queries [30]. Their study reveals that the SQL statements of the code blocks can be automatically analyzed to identify error-prone patterns which can be used in a recommendation system [30].

Above studies analyze one aspect of programming or software development and none of these studies focus on analyzing questions related to logging. In contrast to these studies, the work presented in this paper focuses on analyzing source code logging questions on six StackExhange websites.

3. Empirical study

3.1. Research dimensions and research questions

Table 2 shows three main research dimensions (RDs) and respective research questions (RQs) considered in this work. Following is a brief description of each RD and respective RQs:
RD1: Statistical Analysis of Logging Questions on the Stack Exchange Sites: In RD1, we explore *how* software development communities use Stack Exchange websites for asking logging related issues. For this, we analyze several parameters related to logging questions. We analyze the trend of logging question with accepted answers (**RQ1**), number of answers posted for each logging question (**RQ2**), and time taken by each logging question to get the accepted answer (**RQ3**).

Successful questions (question with an accepted answer) depict satisfaction of the programmer. These trends are essential for development of logging tools and libraries. We conducted this analysis on six websites and hence, it broadens the observation of satisfaction of logging users across various platforms. For an instance, significantly more accepted questions were observed in Database Administrator (DB) than Android (AE). This also gives the sense of how alive is logging today and how much more research and development is required in order to satisfy the needs of programmers dealing with source-code logging

Number of answers per question is the sign of the amount of discussion source-code logging is attracting on these six-websites. Additionally, if a question is attracting a large amount of discussion then it may symbolize the presence of some widely occurring error or some ambiguity faced by the programmers. Study of these cases will aid in developers and researchers in developing tools and methods that would be easy to use and debug. For example, *logcat, alogcat* and *adb* questions invite great amount of discussion in Android.

Time taken to get an accepted-answer to a question corresponds to the time taken to solve user's posted issue. Lesser the time, quicker the solution. If some questions are taking large time in getting accepted-answer, it can depict the presence of some esoteric (lesser known) issues that need to be researched in order to present a palatable solution. If a logging tool or library is associated with large time-taken then concerned developers should intervene with a solution or some version update in order to fix such problems. For example, our analysis shows that questions asked on SE website invite a great amount of

Table 2. Details of research dimensions and research questions

Research dimension	Research questions
Statistical analysis	1. What is the trend of successful and ordinary or unsuccessful questions on logging across years and across Stack Exchange sites? 2. What is the trend of logging question in terms of quantity of answers per question across years and across websites? 3. How much time it takes to get the accepted answer of logging questions?
Programming language analysis	4. How pervasive is software logging related questions on community based Q & A websites across programming languages? 5. What is the distribution of logging questions with respect to different programming language for each Q & A website?
Content analysis	6. What are the main discussion logging topics in various websites? 7. What is the distribution of logging-related tags across various Stack Exchange websites? And how persuasive is the commonality between these tags along various Stack Exchange websites?

discussion. On SE website the user is asking fundamental questions like which methods of better for logging *file* or *database*.

RD2: Analysis of Logging Questions with Respect to Different Programming Languages: In RD2, we analyze trend of logging questions with respect to different programming languages. First, we analyze how pervasive software logging questions are on the community based Q & A websites across programming languages (**RQ4**). Second, we analyze, the distribution of logging questions with respect to different programming language for each Q & A website (**RQ5**).

Analysis of RQ4 provides insight into the development of source-code logging tools and libraries. The results of this RQ will be helpful to analyze the dependency of the programming languages with source-code logging. The results will help in estimating programmer's interest and discussion with respect to various languages. This will help developers to understand emerging trends in programming languages and they would be able to wisely choose a programming language for building logging tools. For example, our results show that maximum number of logging questions are asked in Java and C++. Companies can use this information to build new logging tools.

The RQ5 which is an extension of RQ4. In this RQ, we analyze programming language distribution across six-websites. This would aid developers to choose programming language based upon various environment and platforms. For Example, Server (SF) and super-user (SU) oriented applications suggest a large interest in C-based logging tools while for software engineering, C#, Java, and C++ seems to be a viable option.

RD3: Content Analysis of Logging Questions on the Stack Exchange Sites: We analyze the information present in logging questions. We perform two types of analysis in this: First, we identify the main topics present in the title and description of logging questions (**RQ6**). Second, we analyze the tags associated with the logging questions (**RQ7**).

Results of this dimension provide an overview of most discussed logging topics. This insight will help developers to keep in mind these discussions while developing logging tools and libraries. It would also aid to keep a track of logging-related issues and needs of programmers. In RQ6, we focus on analyzing the content of the post. Answer of this RQ, provides important insight like Android users face logging issues in network connections. Research community can use this information to further improve logging functionality of network related functions in Android OS.

The analysis of RQ7, provide information about cross-discipline logging tools and practices, for example, the transaction log is used in both server environment (SF), Stack Overflow (SO) and database (DB) while event-logging practice is observed in super-user (SU) and Stack Overflow (SO). This knowledge is the use-case for researchers and developers to select logging practices and tools that are compatible with multiple platforms and environments.

3.2. Research method

In this subsection, we describe the research methods followed in this work. There are several Q & A websites such as Stack Exchange [31], Quora [32], where people can post their questions and other people or experts can reply to their questions. In this work, we select Stack Exchange websites for our analysis because it is a network of so many popular Q & A sites. At the time of this study, there were a total of 133 websites present in the Stack Exchange network. The Stack Exchange network consists of websites related to various domains such as software development, tourism, academia. Our aim in this work is to analyze questions related to logging. Hence, analysis of all of these websites is not required and is out-of-scope of this paper. Thus, we carefully selected six technical Q & A websites from all these websites. Following is the criteria and essential properties that we took into account while selecting websites for our study:

Type – Software development/Uses: In this work, we are analyzing questions related to logging. Hence, we select websites related to software development and programming.

Number of users – At least 1000: We select websites having at least 1000 users in order to draw statistically significant conclusions.

Number of questions – At least 1000: We select websites having at least 1000 questions so that we can draw statistically significant conclusions.

Age of the website – At least 2 years old: We select websites having at least 2 years of history. Website which are not so old or are in there beginning phase may not be appropriate for our study as they may not have enough logging questions to infer any statistically significant conclusion.

3.3. Experimental dataset details

Matching to our selection criteria we select following six popular websites that are frequently used by software practitioners. All the six websites are actively used by thousands of users.

Stack Overflow (SO): SO is a Q & A website created for professional and enthusiast programmers [8]. It is created in the year 2008, i.e. $\approx$ 10 years old. At the time of this study, it consisted of $\approx$ 8.2 million users, $\approx$ 14 million total questions and $\approx$ 75 K logging questions.

Server Fault (SF): SF is a Q & A website for system and network administrators [9]. It is created in the year 2009, i.e. $\approx$ 9 years old. At the time of this study, it consisted of $\approx$ 0.3 million users, $\approx$ 0.2 million questions, and $\approx$ 4.2 K logging questions.

Superuser (SU): SU is a Q & A website for computer enthusiasts and power users [10]. It is created in the year 2009, i.e. $\approx$ 9 years old. At the time of this study, it consisted of $\approx$ 0.6 million users, $\approx$ 0.3 million questions, and $\approx$ 1.2 K logging questions.

Database Administrators (DB): DB is a Q & A website for database professionals who wish to improve their database skills and learn from others in the community [11]. It is created in the year 2009, i.e. $\approx$ 9 years old. At the time of this study, it consisted of $\approx$ 0.1 million users, $\approx$ 60 K questions, and $\approx$ 1.1 K logging questions.

SoftwareEngineering (SE): SE is a Q & A website for professionals, academics, and students working within the systems development life cycle [13]. It is created in the year 2010, i.e. $\approx$ 8 years old. At the time of this study, it consisted of $\approx$ 0.2 million users, $\approx$ 47 K questions, and 198 logging questions.

Android (AE): AE is a Q & A website for enthusiasts and power users of the Android operating system [12]. It is created in the year 2010,

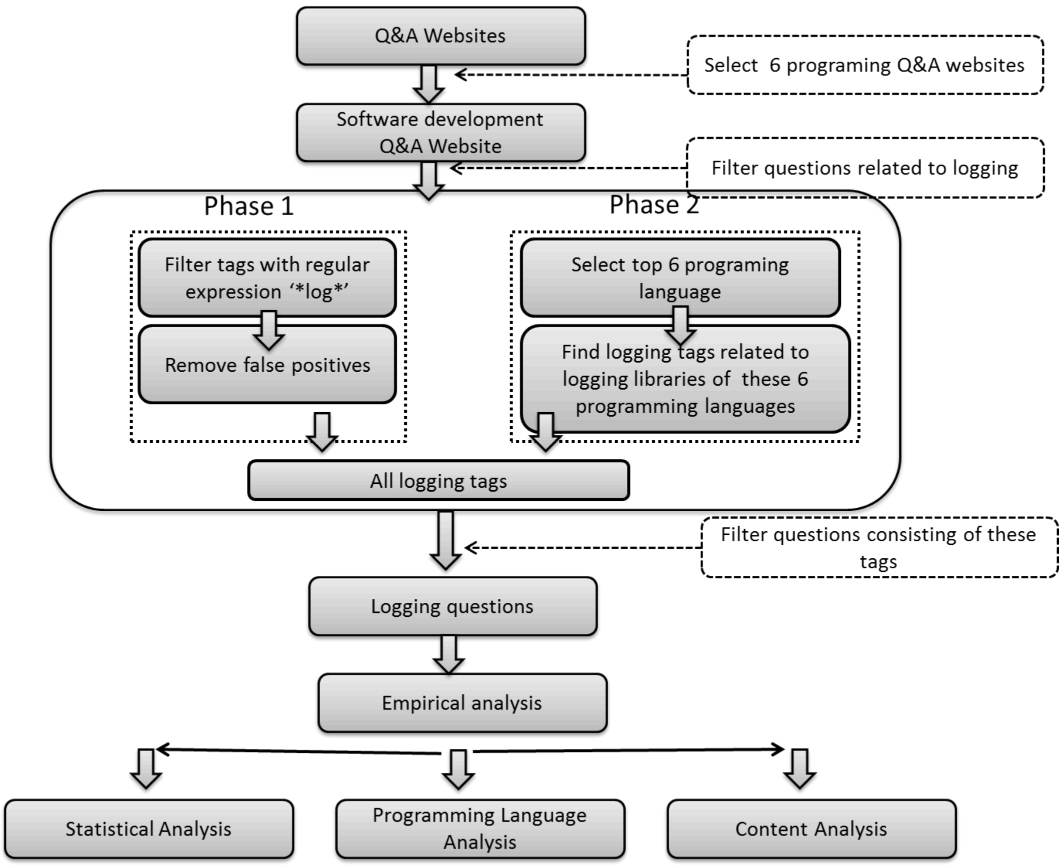

Q&A Websites
Select 6 programing Q&A websites
Software development Q&A Website
Filter questions related to logging
Phase 1
Phase 2
Filter tags with regular expression '*log*'
Remove false positives
Select top 6 programing language
Find logging tags related to logging libraries of these 6 programming languages
All logging tags
Filter questions consisting of these tags
Logging questions
Empirical analysis
Statistical Analysis
Programming Language Analysis
Content Analysis

Figure 1. Research method followed in this study

i.e. ≈ 8 years old. At the time of this study, it consisted of ≈ 0.1 million users, ≈ 46 K questions, and 183 logging questions.

3.4. Dataset preparation

In this subsection, we describe the steps that we used to extract the relevant dataset for our study. For this study, we have used the data dump provided by Stack Overflow community. This dataset is in XML format and consists of details of all the questions asked by users. For each website, it provides 7 files: badges.xml, comments.xml, posts.xml, posthistory.xml, user.xml, votes.xml, postlinks.xml. For this study we have used **posts.xml** file. This file consists of information each post. For example, if for a give question there are three answers, then a total of four post will be included in this file. This file consists of information like, title of the questions, description of the questions, date on which the questions asked, etc. Next, we extract all the logging questions. Manual identification of all the logging questions can be a tedious task. Hence, we adopt a method to automatically find the logging questions. We use tags assigned to questions to identify logging questions. The Stack Exchange community assigns a set to tag to each question. These tags are chosen carefully to describe the domain of the question. We use a **2-phase** method to select all the logging tags. Figure 1 present the main steps of our **2-phase** method. Below we describe our **2-phase** method:
Phase 1: In phase 1, we define a regular expression, i.e. **log** to retrieve all the logging tags. We notice that this approach is very effective in finding logging tags, as we were able to retrieve several logging tags using this approach. For example, we are able to retrieve tags such as

transaction-log, *syslog*, *syslog*. However, we notice that this approach results in lots false positives also. For example, tags like *'login'*, or *'logins'* were also outputted. Hence, we manually remove false positives from the dataset.

Phase 2: We noticed that the phase 1, is not able to retrieve logging questions because the regular expression used in the phase 1 is not able to retrieve tags such as *SLF4J*. Hence, we decided to add these kinds of tags manually. We select top 6 programming language from the tiobe index, i.e. Java, C, C++, Python, C#, and JavaScript. We perform an exhaustive Google search to identify all the logging libraries used for these six programming languages. For example, *Log4J* and *SLF4J* for Java, *Log4C* for C, C++. We add tags related to these libraries in our list.

Using process followed in Phase 1 and in Phase 2, we retrieve a total of 169 tags.We extract all the questions consisting of any of these tags. We extracted 82199 logging from these six websites. We made all our dataset publicly available to allow replication of the results by software engineering research community (https://github.com/newtein/StackExLogging). For each website, we compute the percentage that logging questions have with respect to total questions. Table 3 shows that SF and DB have the highest percentage of logging questions. This table also shows that a large number of questions are asked on logging.

3.5. Research contributions

In context to work done in literature, in this work, we perform the first study (to the best of our knowledge) of logging questions on six popular Stack Exchange websites with respect to three dimensions. We identify several RQ's related to statistical and content analysis of logging questions. We answer each RQ by conducting empirical analysis on more than 82 K logging questions.

3.6. RD1: Statistical analysis

In this subsection, we present the results of various RQ's related to statistical properties of logging questions. This research dimension provides information about how the behavior of programmers is changing over time. Statistical trends are observed from posted questions and answers. This dimension of research may not have a direct application for a user but it is in-fact quite essential to understand how source-code logging

Table 3. Experimental dataset details of Stack Exchange website – Stack Overflow: SO, SuperUser: SU, Server Fault: SF, DBA: DB, SoftwareEngineering: SE, Android: AE

Field	SO	SU	SF	DB	SE	AE
Total Number of Unique Users	8287574	630516	346259	114789	241851	154687
Total questions	14995834	363915	252963	60948	47362	46559
Total questions with accepted answer	8034235	154322	125601	29400	27762	13316
Total logging questions	75185	1275	4227	1131	198	183
Total logging questions with accepted answer	39674	541	2163	555	110	50
Percentage of logging questions to total questions	0.19	0.14	0.62	0.78	0.10	0.17
Timestamp of the First Question	8/1/2008	7/15/2009	4/30/2009	10/22/2009	9/27/2010	9/1/2010
Timestamp of the Last Question	12/3/2017	11/30/2017	12/1/2017	12/1/2017	11/19/2017	11/28/2017

is evolving over the years? This gives a sense of how alive is logging today and how much more research and development is required in order to satisfy the needs of programmers dealing with source-code logging.

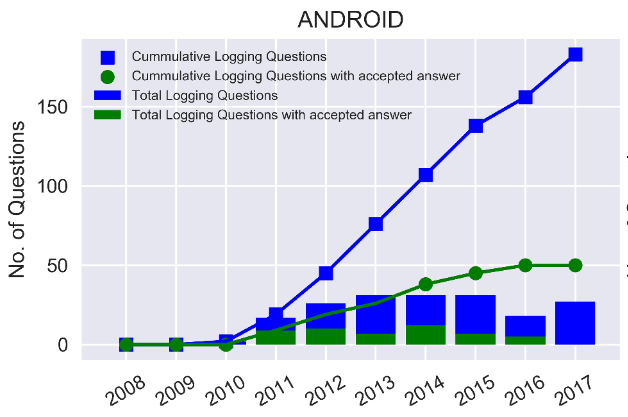
3.6.1. RQ1: What is the trend of successful and ordinary or unsuccessful questions on logging across years and across Stack Exchange sites?

Motivation: On Stack Exchange sites a question can receive multiple answers. The user who has asked the question can review these answers. If he is satisfied with one of these answers, he can mark that answer as *accepted* [33]. However, if none of these answers, answer the question correctly, the user has the right to not select any of these answers as accepted. Each question can have only one *accepted* answer. In the literature [14], Stack Exchange questions are classified into three categories: *successful* (questions with accepted answer), *ordinary* (questions that have answers but none of them is accepted) and *unsuccessful* (questions that do not have any answer). In this RQ, we analyze the trend of logging questions and logging questions with accepted answers on various Stack Exchange websites. Accepted answers are indicators of quality of responses to questions. Accepted answers shows that questions on logging are receiving helpful responses. We believe that identifying number of logging questions that are successful can be beneficial in identifying the behavior and satisfaction of software development community towards logging.
Approach: In this RQ, we extract total logging questions and logging questions with accepted answers (i.e. successful logging questions) for each of the six websites. We extract this data for all the years from 2008 to 2017. Using this information, we plot histogram showing the total number logging questions and total successful logging questions for each of the six website. We also plot cumulative logging questions and cumulative successful logging questions for all the six websites.

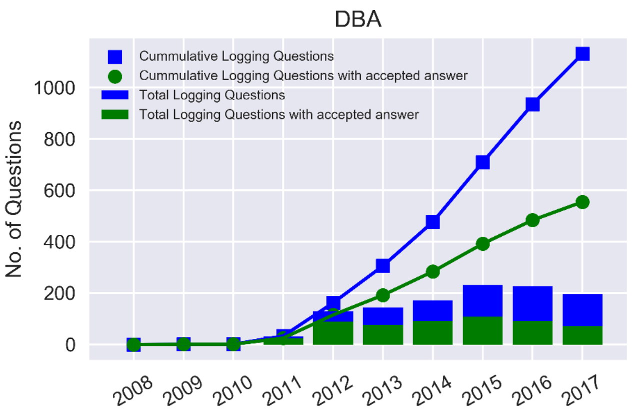
Results: Figure 2 shows the histogram of total logging question and successful logging questions. It also shows the trend of cumulative logging questions and cumulative successful logging questions. From Figure 2, we draw several interesting observations. First, it shows that irrespective of the website logging questions occur consistently across all the websites. For example, on SU website users had asked 50–177 questions in each year between 2009 and 2017. Second, we observe that the frequency and intensity of questions differ across the websites. For example, a total of 75185 logging question are asked on SO whereas 1131 logging questions are asked on DBA in the years 2008–2017. This huge difference in the number of logging question between DBA and SO does not necessarily means that database have less logging issues. It can be also be due to difference in the popularity and user base of the websites. For example, the SO website has much bigger user base and is much more popular than other Stack Exchange websites, and hence, has much more logging questions as compared to other sites. Third, we observe that there is no trend (increasing or decreasing) in number of logging questions asked over the years for all the websites. Fourth, we observe that all the websites have a large number of successful logging questions. For example, 33–78% of logging questions have received an accepted answer on the SO website.
RQ1 conclusions: Logging is an important concern that occurs frequently in different domains. We observe a large number of successful logging questions. However, we do not observe any trend in terms of frequency of total logging questions and successful logging questions across the years for any website.

3.6.2. RQ2: What is the trend logging questions in terms of quantity of answers per question across years and across websites?

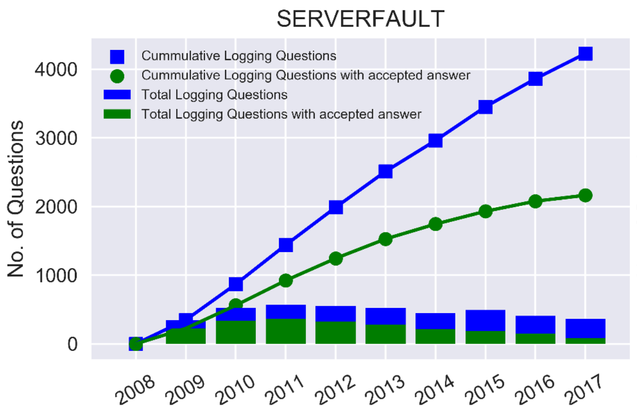
Motivation: In this RQ, we analyze the number of answers posted for each logging questions. On Stack Exchange sites, users can post any number of answer to each questions. We consider answer



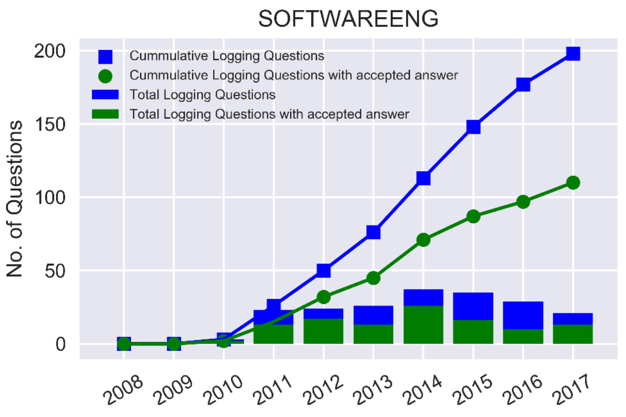
(a) AE



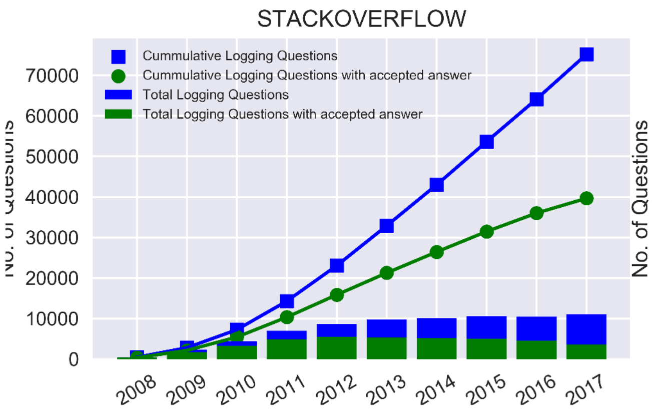
(b) DB



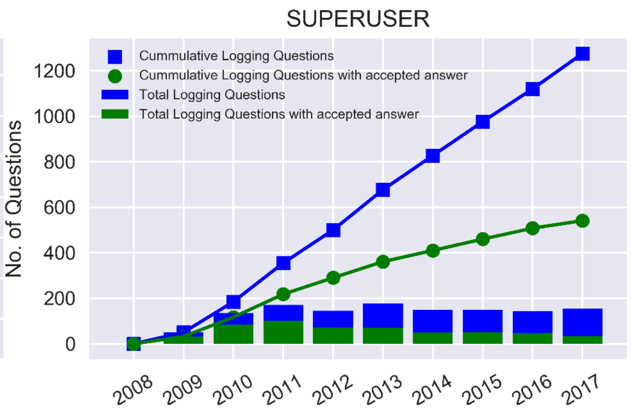
(c) SF



(d) SE

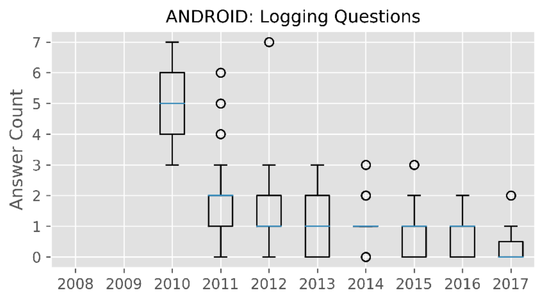


(e) SO

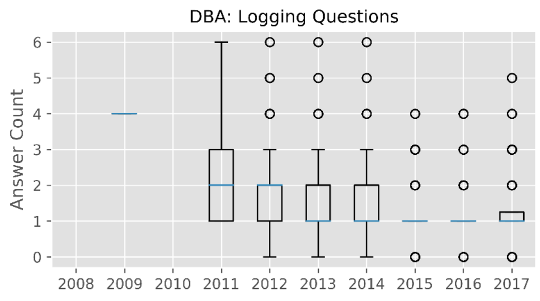


(f) SU

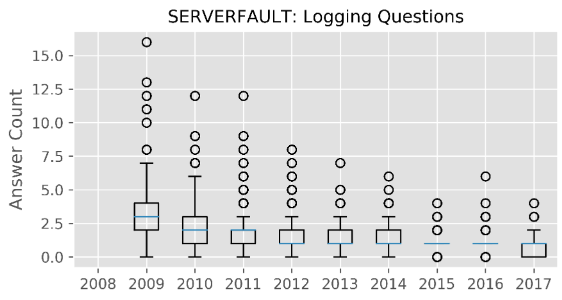
Figure 2. Distribution of logging questions with accepted answers



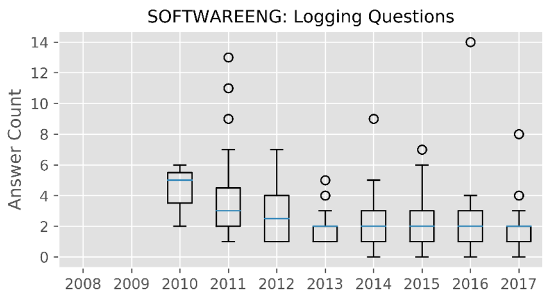
(a) AE



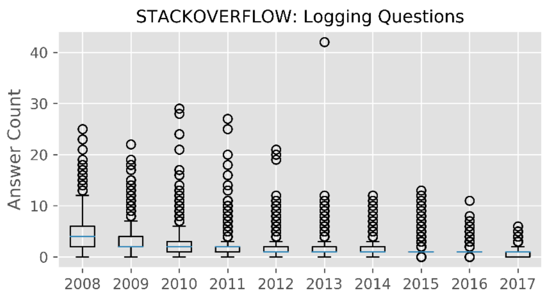
(b) DB



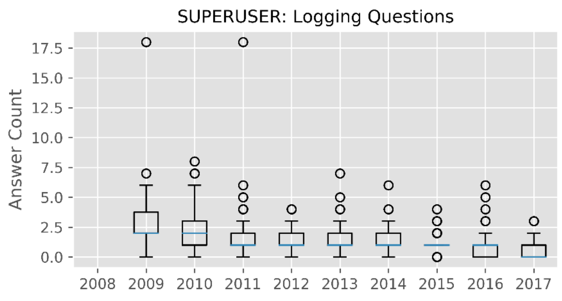
(c) SF



(d) SE



(e) SO

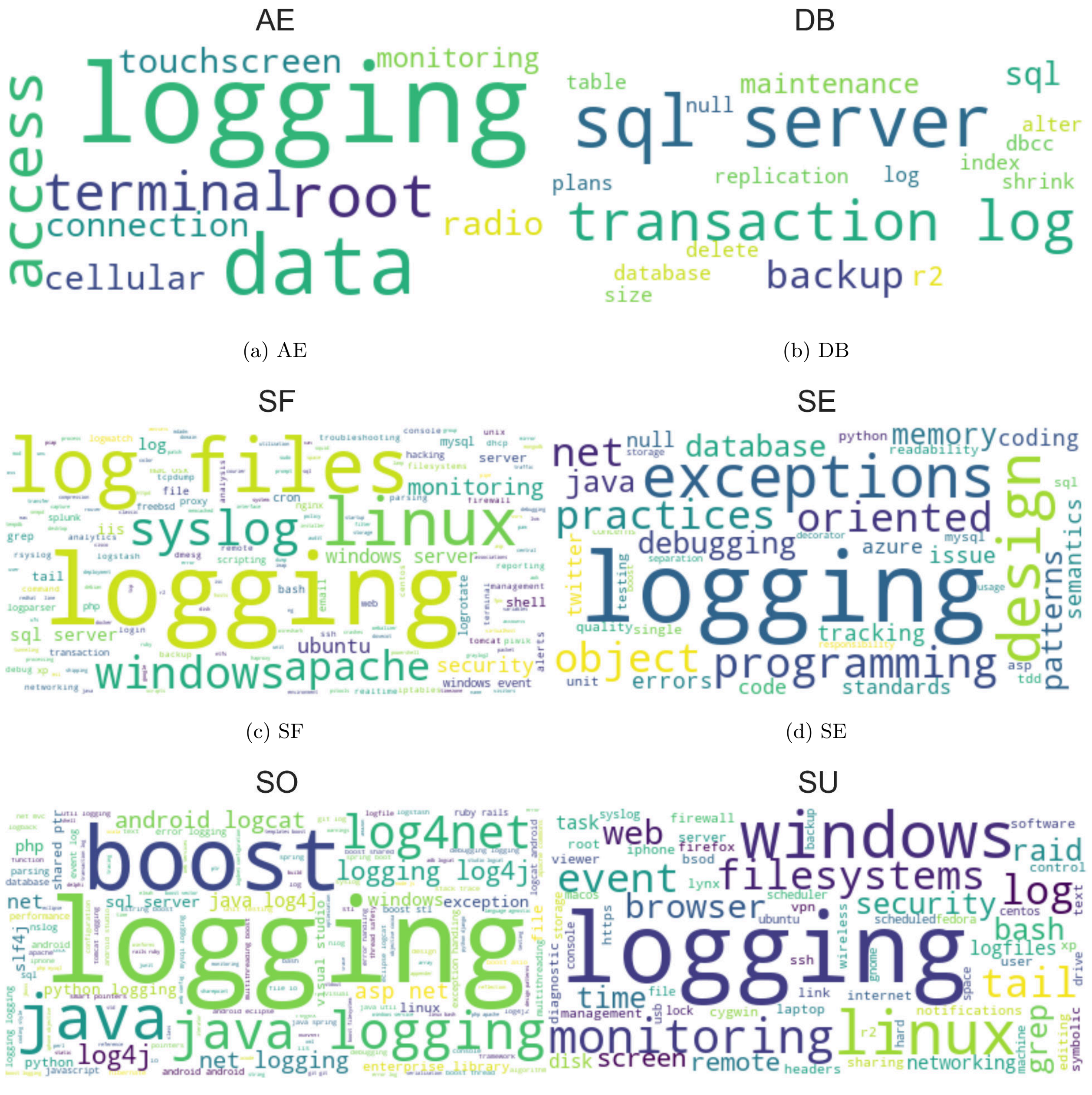


(f) SU

Figure 3. Box plot showing answer count of all the logging questions

Table 4. Percentage of the questions that received ≥5 answers – Stack Overflow: SO, SuperUser: SU, Server Fault: SF, DBA: DB, SoftwareEngineering: SE, Android: AN

SO	SU	SF	DB	SE	AN
2.8	2.35	3.76	1.06	**11.61**	2.18



(a) AE (b) DB

(c) SF (d) SE

(e) SO (f) SU

Figure 4. Word cloud of tags of logging questions that received more than 5 answers

count posted for each question as a measure of **discussion**. Increase in number of answers can be an indication towards increase in discussion required for logging questions.

Approach: To answer this RQ, we compute number of answers received for each logging question for each year for all the six websites. Using this information, we compute descriptive statistics such as Quartile-1 (Q1), Median, Quartile-3 (Q3), Min and Max and create box-plots. We compute descriptive statistics to gain insight on the data characteristics and its basic features.

Results: Figure 3 shows box-blot of number of answers received for each logging question for all the websites. We study the central tendency of the data in-terms of the median values. The median values of the answer count for the sites AE, DB, SF, SE, SO and SU in the year 2014 are 1, 1, 1, 2, 1, and 1, respectively. The box-plot in Figure 3 reveals the dispersion in the data which is the spread of the values around the median. We draw several interesting observations from the Figure 3. First, we notice that for all the websites, the median value of the number of answers received for each question is higher in the initial years (2008–2011) as compared to later years (2012–2017). For example, for SE website the median values of answer count is 5 (2010), 3 (2011), 2.5 (2012) and 2 (2013–2017). For this outcome, one reason can be that old questions receive more answers over the period of time.

Second, we observe presence of several outliers in all the websites. We show the outliers in Figure 3 using dots so that they are clearly visible and displayed separately and do not exaggerate the range values. We observe that the SO website has the highest number of outliers as compared to any other website. To get insight about the questions receiving a large number of answers, we further analyze logging questions that received ≥5 answers. Table 4 shows the percentage of logging questions that receive ≥5 answers. Our analysis shows that ≈1.06–11.61% of questions in all the websites received ≥5 answers. It is interesting to find that the SE website has the highest percentage of questions that received ≥5 answers.

Third, we analyze tags assigned to logging questions that received ≥5 answers. We build word cloud of tags associated with these questions (refer to Fig. 4). We observe the word cloud of each website highlights a different set of tags. For example, in Android logging questions are related to *logcat*, *alogcat* and *adb*. In DB website, all the 12 questions are related to *transaction-logs*. The SE websites word-cloud highlights tags like *exception, practice*. The logging questions in SE website are related to *logging practices*. The most discussed logging questions on the SO website are related to logging libraries such as *log4net, log4j, SLF4J*. Table 5 provide more details about the questions that we analyzed.

RQ2 conclusions: Approximately 1.06–11.61% of all the logging questions invite a great amount of discussion.

Table 5. Analysis of the questions that received ≥5 answers: Stack Overflow: SO, SuperUser: SU, Server Fault: SF, DBA: DB, SoftwareEngineering: SE, Android: AN

S.No.	WB	Tag	ID	Context
1	AE	root-access	157	A rooted device can monitor the **logcat (a command line tool that dumps a log of system messages)** stream on the phone for this he needs root access.
2	AE	touchscreen	13992	is there an existing app that could be installed and could **record touch interactions** on the background?
3	AE	Android logging	14430	Android logging can be viewed and examined by use of **adb logcat(android debug bridge it can control device over USB from a computer)**, **alogcat(software testing tool that control full control over intents)**, **logcat extreme (user interface that records and monitors logcat)**.
4	AE	data connection, data-monitoring, celluar-radio	35702	**Logging information about data-connection**, rate along with the location can be monitored by using phone's radio without sending any additional data which is available through logcat or through network buffer on user's phone using alogcat or adb.

Table 5 continued

S.No.	WB	Tag	ID	Context
5	DBA	sql-server, transaction-log, dbcc, database-size, shrink	41215	Shrinking of database on SQL server using DBCC while **transaction log** showed that no transactions where open can be done using log_reuse_wait_desc query and further sp_removedbreplication query to remove replication related objects.
6	DBA	sql-server, transaction-log, r2, backup	45876	The only difference between full backup and copy-only-full-backup is that the full backup does not break up the differential log chain while neither of them breaks the **transaction log** chain nor they truncate the transaction log file.
7	DBA	sql-server, transaction-log, backup, delete	13757	Prior taking the backup, to safely remove the SQL server transaction log file use sp_detach_db procedure. This procedure make sure that SQL server records the fact that database was shut down cleanly.
8	DBA	sql-server, backup, transaction-log	162628	Taking a **transaction log** backup and truncating the log and then deleting the **transaction log** backups this would only work if the tool is using its own tracking to know which log files are to be restored for the proper functioning of log-chain.
9	DBA	sql-server, transaction-log, maintenance	12474	During the maintenance of **transaction-log** file it should be make sured that there are no error in backing up transaction log otherwise file size would grow and system would run out of space.
10	SF	monitoring	1845 24428 53000 53699 53894 437369	Here monitoring of different types of **log files, systems** and their different features is being made.
11	SF	syslog, logfiles	96720 42527 49042 62687	Different tools like logrotate, splunk linked to **syslog log** files are studied.
12	SF	Apache	322116 355311	To catch all access log with Apache virtual hosts, to write useful awk and grep scripts for Apache logs and such other log based features based are provided by Apache.
13	SE	exceptions	15502 20109 130250 272771 306032	Different features to handle different types of **logging exceptions**.
14	SE	object	82, 499 230, 131	It provides features like **best design perspective for logging**, need of logging while doing TDD(Test-driven development).

Table 5 continued

S.No.	WB	Tag	ID	Context
15	SE	debugging	84, 301 225, 903	It explains use of the concept like **timestamping, maintaining transactions log and logging** for the purpose of debugging.
16	SE	design	27, 595 782, 499 153	It provides different design perspective for **best logging practices**.
17	SE	programming	2, 727 713, 729 415, 500	It tells us to write **efficient programs** which make significant use of logging.
18	SU	monitoring	103, 222 143, 658 226	It explains **monitoring concepts of logging** from the network perspective.
19	SU	filesystems	22, 674 420, 321 236, 100	It **explains filesystems with respect to logging issues**.
20	SU	Linux	106073 222912 226744 330590 351387	It helps in **logging different processes, files** in operating system Linux (Ubuntu).
21	SU	Windows	153 106073 145086 219401	It helps in **logging different processes and files** in OS Windows.
22	SO	log4net	192456 756125 50599689 50591008	The **Apache log4net** library is a tool to help the programmer output log statements to a variety of output targets.
23	SO	log4j,	1140358 12532339 728295	**Apache Log4j** is a Java-based logging utility.
24	SO	logcat	7959263 2250112 3280051 19897628	**Logcat** is a command-line tool that dumps a log of system messages, including stack traces when the device throws an error and messages that you have written from your app with the Log class.
25	SO	boost	34362871 17844085 39247778 34394896	Boost is a set of libraries for the C++ programming language that provide support for tasks and structures such as linear algebra, pseudorandom number generation, multithreading, image processing, regular expressions, and unit testing.
26	SO	SLF4J	11916706 7421612 8965946 14024756 11639997	**SLF4J** or Simple Logging Facade for Java provides a Java logging API by means of a simple facade pattern.

3.6.3. RQ3: How much time it takes to get the accepted answer for logging questions?

Motivation: In this RQ, we compute time taken by the logging questions to get an accepted answer. We believe that answer to this RQ can be beneficial in identifying type of logging issues that are most time consuming. For example, server related issues can be more time consuming as compared to others.

Approach: We perform three types of analysis to answer this RQ. First, we compute the average time taken by the logging questions to get accepted answer. Second, we compute average time taken by the logging questions to get accepted answer for each year separately for all the six websites. We also compute standard deviation for both the analysis. Third, we create box plot of the time taken by the logging questions to get the accepted answer. We compute both average time graph and box-plot because average computation is affected by outliers and can give mis-leading results, whereas in box-plot analysis all the outliers are clearly visible.

Results: Figure 6 shows the average acceptance time for all the logging questions for all the years combined. Figure 6 shows that mean time to get accepted answer for SU and SF websites are much higher as compared to other websites. The mean time of acceptance of SU and SF websites in 36761.93 (in hours) and 26034.32 (in hours), respectively. The standard deviation of SU (198665.32) website is much higher as compared to other websites, i.e. AE (76106.25), DB (119476.32), SF (159452.76), SE (167535.09), and SO (138574.83).

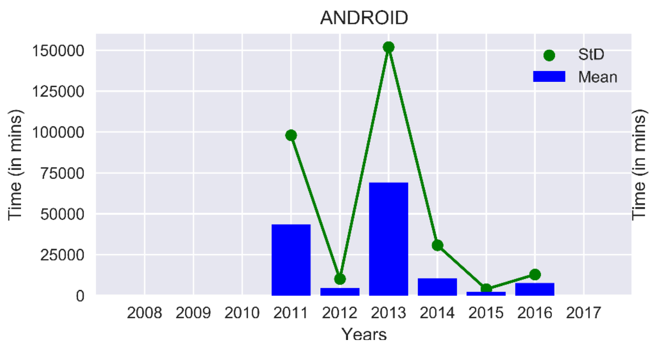
Figure 5 shows the average acceptance time for all the logging questions for all the years separately. We observe that for each website whenever there is an increase in the mean acceptance time, there is corresponding increase in the standard deviation, which indicates presence of potential outliers, i.e. questions that took a large amount of time to get the accepted answer in all the websites. Additionally, we observe presence of several questions which took almost 0 time in getting accepted answer. For example, SO has 156 questions that took 0 second to get an accepted answer. Figure 5d shows that the mean time taken by the SE questions is very less for all the years. In year 2012, there are some question in SE that took much longer to get the accepted answer. The SE website invites questions on programming practices. If there is some question which is taking a large amount of time, it can indicate some fundamental programming issue or concern with logging in which software developers are facing problem. We analyze five questions on SE website that took large amount of time to get accepted answers (refer to Table 6). Following is the detailed analysis of two such questions:

In question 1 (id: 291757), the user is asking about "a better method to handle precondition and logging". The experts suggested the user to use *throw* and *assert* statements. Following a snippet of expert comment:

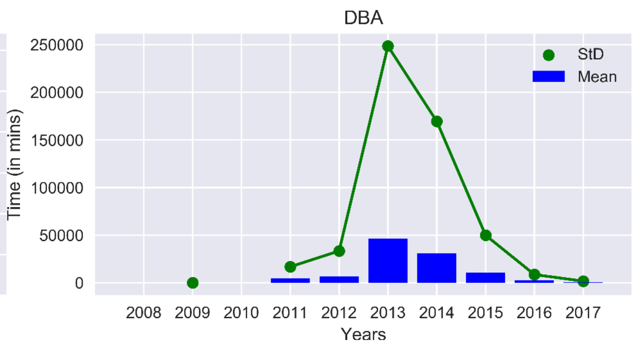
> When implementing this, you should rarely decide on how to handle the error, at the place where it occurs; instead, you should throw an exception and let client code decide.

In question 2 (id: 208471), the user was asking a fundamental question about a better method between *file* or *database* to use for logging. The expert suggested the user to use file as he was not needing any complex processing of the log used. Following is the snippet of expert comment:

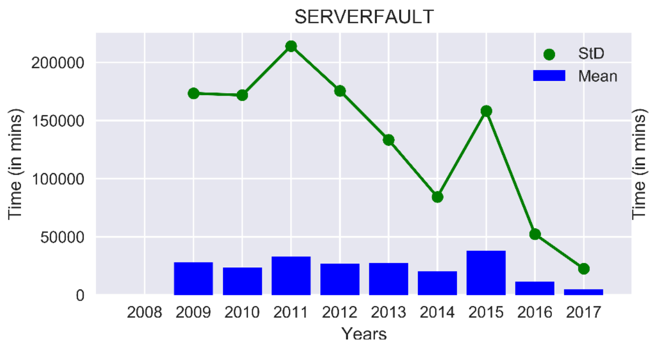
> Both options seem valid to me. In such cases, a useful rule to apply is to do the Simplest Thing That Could Possibly Work. Text files are easier to get started with and are expected to work reasonably well at least in the beginning. Once requirements arise that are better satisfied using a database, it will be trivial to import them. Using this strategy, you postpone design decisions as long as possible (but not longer than that). As such, you don't do unnecessary work. When, if ever, it will be needed, you will have a much better understanding of what exactly it is you need. Hence, you are more likely to build the Right Thing and not waste time building the Wrong Thing.



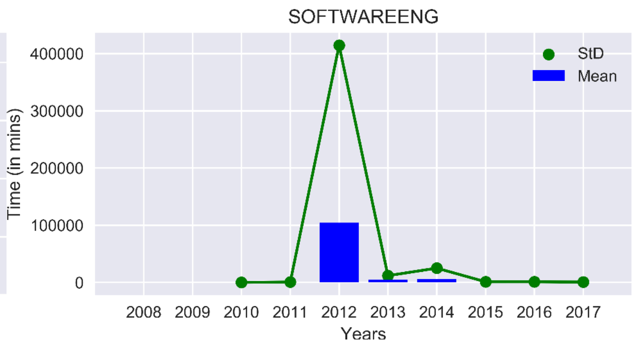
(a) AE



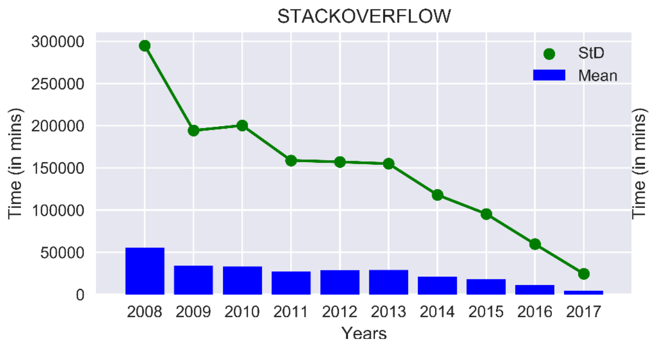
(b) DB



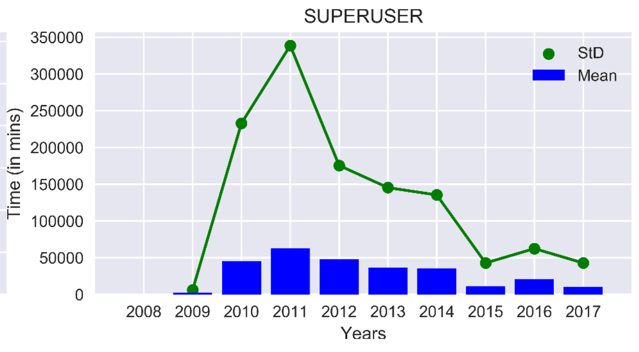
(c) SF



(d) SE



(e) SO



(f) SU

Figure 5. Mean time to get accepted answer for logging questions

Figure 7 shows the box plot of the time taken by the logging questions to get the accepted answer. Figure 7 shows that the median acceptance time of logging questions on the AE website is much higher as compared to that of other websites. For example, the median acceptance time (in minutes) for logging question on the AE website is 3093.39 (2011), 113.2 (2012), 4475.32 (2013), 292.30 (2014), 790.49 (2015), 277.74 (2016) whereas for the SO website is the median acceptance time is 46.96 (2011), 65.41 (2012), 78.43 (2013), 111.94 (2014), 139.91 (2015), 151.86 (2016). There can be several reasons for this kind of behavior for example may be AE logging questions are more complex as compared to other logging questions or there can be lack of user participation of the AE website as compared to other websites.

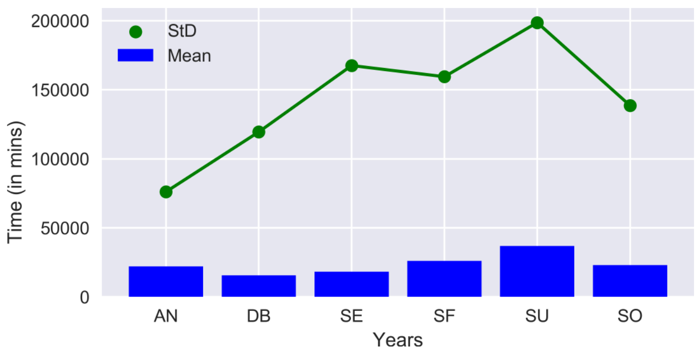


Figure 6. Combined mean

RQ3 conclusions: The *mean time* to get accepted answer for logging questions on SU and SF websites are much higher as compared to other websites, whereas, the *median acceptance time* of logging questions asked on AE is much higher as compared to the other 5 websites.

3.7. RD2: Programming language analysis

This research dimension provides an insight into the development of source-code logging tools and libraries. Results would be helpful to analyze the dependency of programming language with source-code logging across various platform. Results estimate programmer's interest and discussion with respect to various languages. This will help developers to choose a programming language for developing logging tools based upon various environment and platforms. For Example, Server (SF) and super-user (SU) oriented applications suggest a large interest in C-based logging tools while for software engineering, C#, Java, and C++ seems to be a viable option.

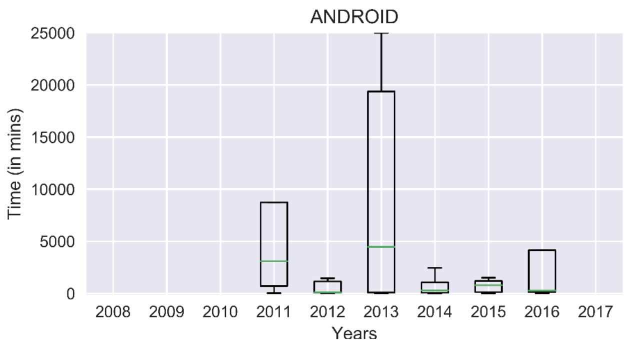
3.7.1. RQ4: How pervasive is software logging related questions on community based Q & A websites across programming languages over the years?

Motivation: In this RQ, we analyze the distribution of logging question in different programming languages over past several years. Logging frameworks for various programming languages can vary in-terms of their capabilities and performance with respect to features such as type-safety, thread-safety, flexibility and portability. Our objective is to gain insights on the quantity of questions asked on logging frameworks for multiple programming languages. We believe that answer to this RQ, can be beneficial in identifying the programming language(s) in which software developers face most of the logging issues. It can also be beneficial in identifying the languages in which logging interest is increasing or decreasing. Answer to this RQ, can be used to tune future logging automation or prediction tools.

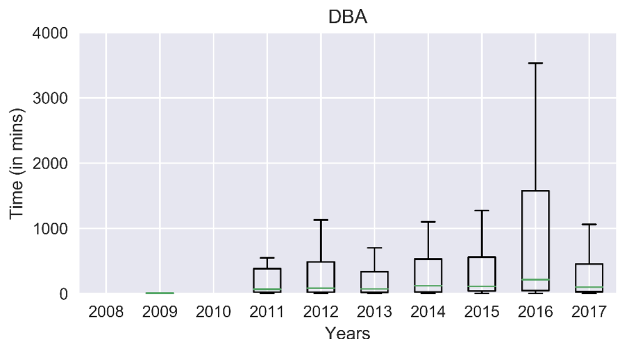
Approach: To answer this RQ, we select top 6 programming languages: C, C++, C#, Java,

Table 6. Top 5 most time consuming questions on the SE website

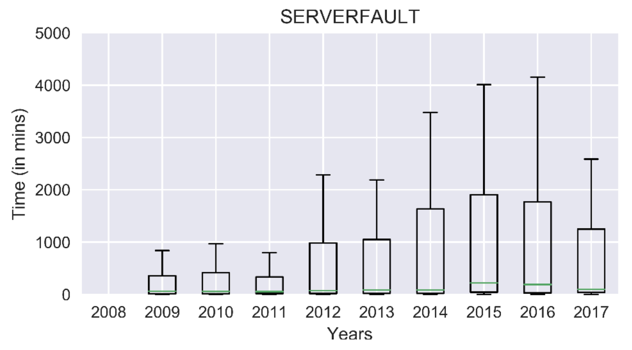
S.No.	Question Id	Answer Id	Question	Time (in minutes)
1	291757	292108	Better way of handling pre conditions and logging	4 647.95
2	208471	209261	Is SQLite a sensible option for data logging?	10 154.03
3	220557	223273	Finding patterns in logs	44 275.5
4	232143	244595	Strategy to store/average logs of pings	130 175.27
5	149346	298292	Logging asynchronously – how should it be done?	1 761 970.55



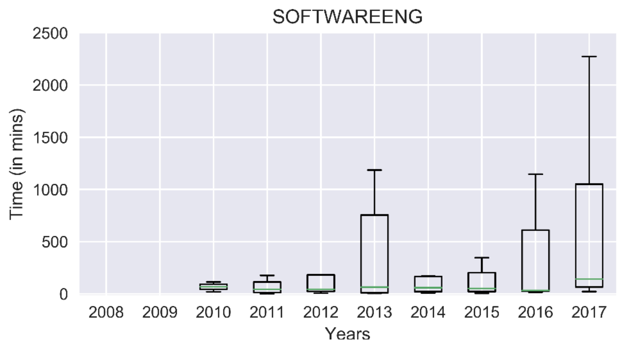
(a) AE



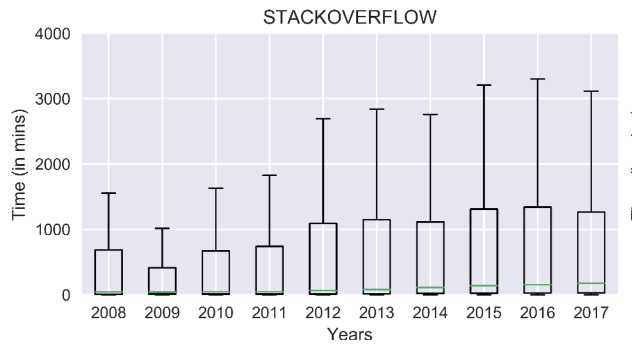
(b) DB



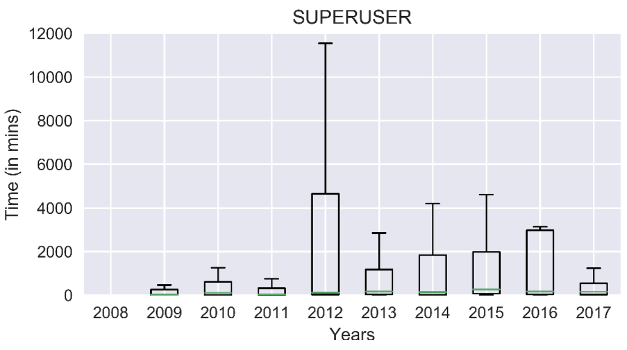
(c) SF



(d) SE



(e) SO



(f) SU

Figure 7. Box Plot showing time to get accepted answer for logging questions

Python, and JavaScript. Next, we extract all the logging related questions for these six programming languages. We collect data from all the six websites considered in this work, i.e. SO, SU, SF, DB, SE, and AN. We extract these questions in two steps:

– First, we manually identify the tags related to popular logging libraries used in these six programming languages. Table 7 shows the list of tags that we identified. We select all the questions consisting of any of these tags. Logging questions are then assigned to their respective programming languages.
– Second, we select all the questions that consist of any of the programming language tag, i.e. 'C', 'C++', 'C#', 'Java', 'Python', 'JavaScript'. From these questions, we filter all the questions that consists of any 'logging

Table 7. Tags that we used to extract questions related to programming languages

Programming language	Tags
Java	Log4J, SLF4J, tinylog, logback, Apache Commons Logging, google-cloud-java, commons-logging, jboss-logging, syslog4j, otroslogviewer, log4j, log4j2, log4jdbc, jul-to-slf4j, hyperloglog.java, java.util.logging
C	Log4C, sclog4c, syslog, zlog, zf_log, log4c
C++	glog, log4cplus, pantheios, boost::log, easylogging++, log4cxx, boost, boost-log, boost-logger, boost.log, spdlog, log4cpp
Python	pygogo, Logbook, google-cloud-python, django-logging, logger-python, hyperloglog.python, unified-log, auth.log, graylog, graylog2
C#	log4net, NLog, Enterprise Library, Common.Logging, log4net-configuration, log4net-appender, log4net-filters
JavaScript	js-logging, Log4js, log4JavaScript, JSNLog, Node-Loggly, Bunyan, Winston, Morgan, Angular-Loggly, loglevel, jsnlog, log-level, logsene-js, node-nslog, truncate-logs-js

tag'. The logging questions are then assigned to the respective programming language.

Results: Figure 8 present the number of logging questions asked in each year (2008–2017) for all the six programming language considered in this work. In this Figure, y-axis represents the number of logging questions asked in each year. We kept the y-axis scale same for all the programming languages for better visualization of logging questions trend across the website. We draw several interesting observations from the Figure 8. First, it shows that users have asked the highest number of logging related questions for Java and C++. A total of 51723 logging questions are asked on these websites out of which 73% (i.e. 37935) questions belong to Java and C++. Second, it shows that the number of logging questions are increasing for Python and JavaScript. For example, the number of logging questions asked in Python are : 11 (2008), 66 (2009), 149 (2010), 247 (2011), 323 (2012), 465 (2013), 512 (2014), 571 (2015), 701 (2016), 789 (2017). We also observe an increasing trend for logging questions in Java (except in the year 2016). Whereas, we observe a decreasing trend for logging question in C++ and C# after the year 2013.

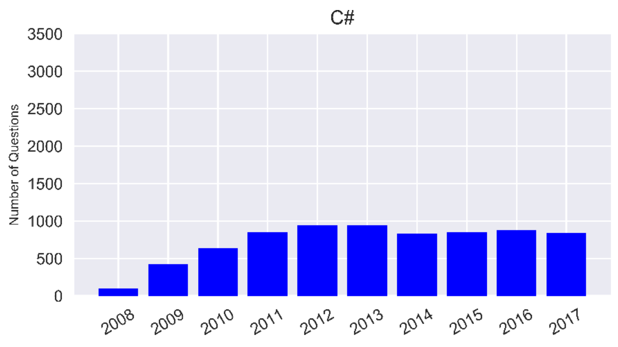
RQ4 conclusions: A majority of logging questions belong to C++ and Java programming language. The trend of number of logging questions is increasing for Java, Python, and JavaScript, whereas it is decreasing or constant for C, C++, C#.

3.7.2. RQ5: What is the trend of logging questions with respect to different programming language for each Q & A website?

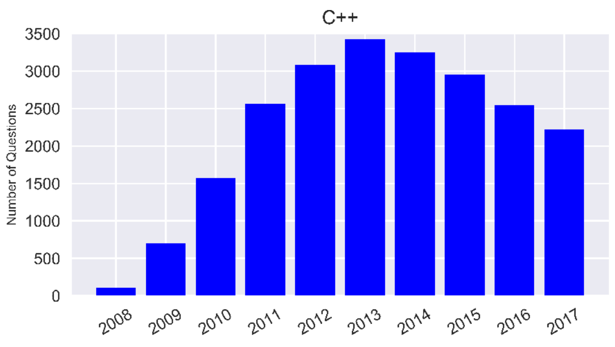
Motivation: In this RQ, we analyze the distribution of logging questions with respect to different programming languages for different websites. Each website represents a specific domain. Analysis of the logging questions asked in different programming language with respect to different website can be beneficial identifying the programming language in which most of the questions arise on that domain.

Approach: To answer this RQ, we compute total number of logging questions asked in each year on each website. For each website, we grouped the logging questions with respect to each programming language.

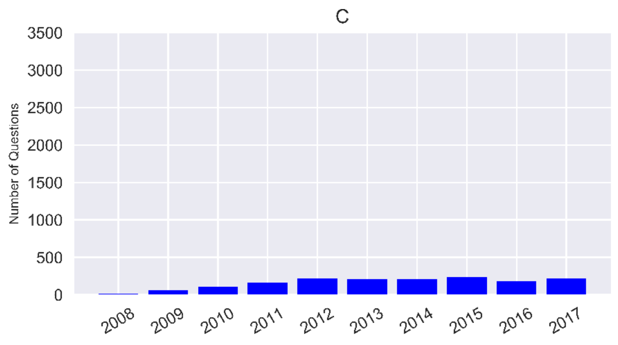
Results: Figure 9 shows the programming language wise trend of logging question with respect to each website. The results of the RQ4, show that Java, C++ and C# are dominant programming language in which most of the logging questions are asked. However, results of the RQ5 show that different programming language show



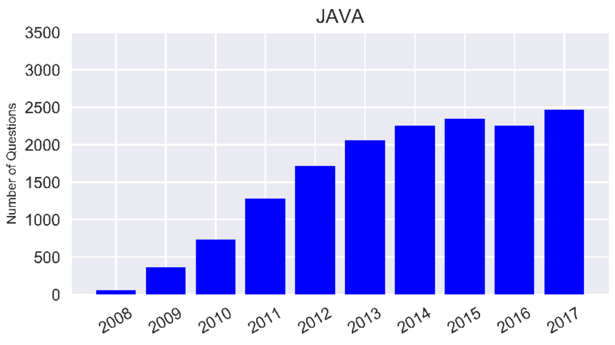
(a) C#



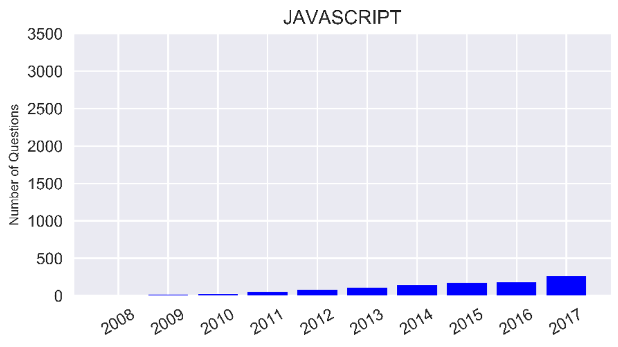
(b) C++



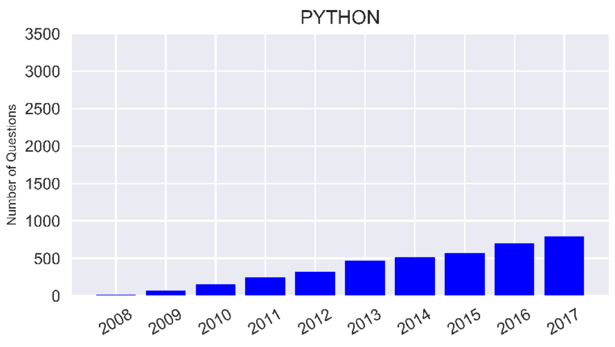
(c) C



(d) Java



(e) JavaScript



(f) Python

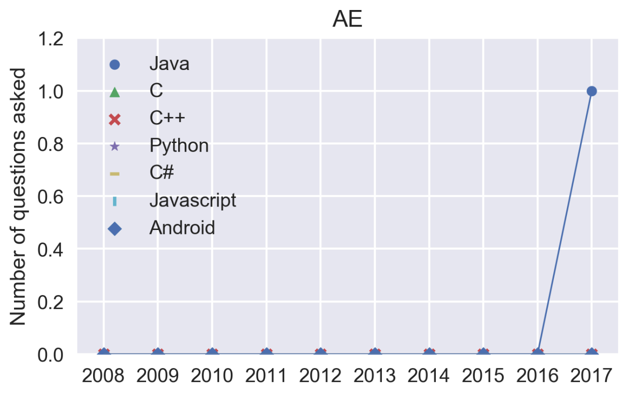
Figure 8. Count of logging questions for various programming languages

dominance (language in which the highest number of logging questions are asked) for different website. For example, for the SO website C++ and Java are dominant language whereas for the SF and SU website, 'C' is the dominant programing language. For the SE website, we did not observe any particular trend with respect to different programming languages. The AE and DB websites consists of very few logging questions and hence, we not able to extract any significant insight.

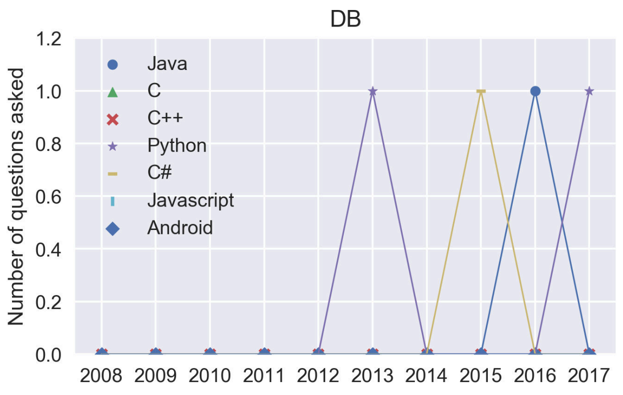
RQ5 Conclusion: Different websites have different programming language that is dominant. For the SO website C++ and Java are dominant language whereas for the SF and SU website, 'C' is the dominant programming language.

3.8. RD3: Content analysis

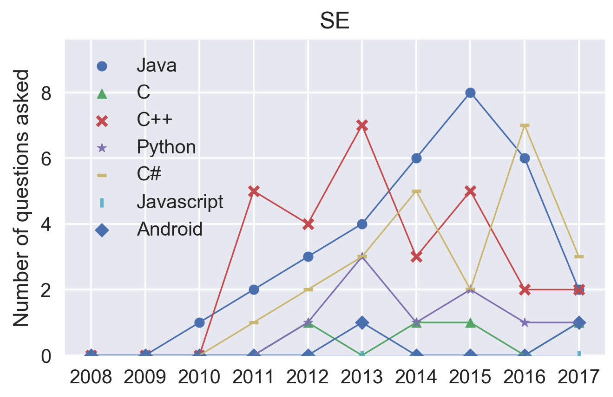
Source code logging exists in various forms and across various disciplines. Results of this dimen-



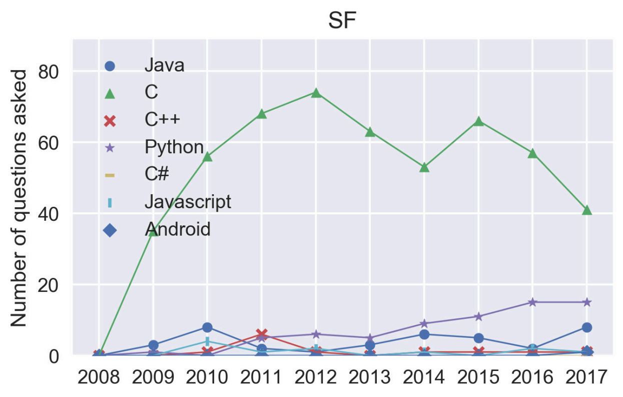
(a) AE



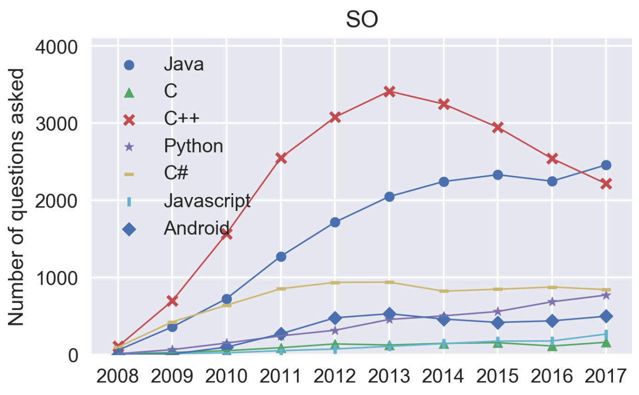
(b) DB



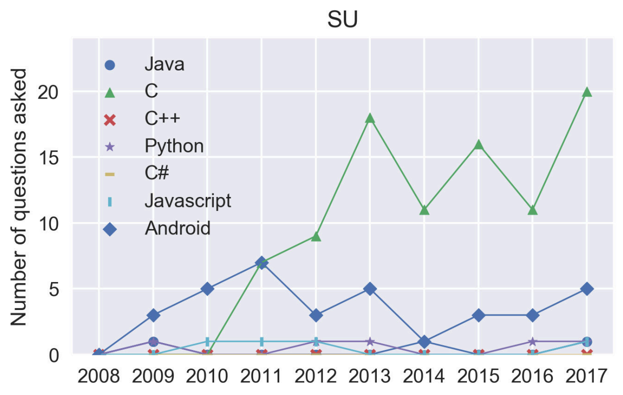
(c) SE



(d) SF



(e) SO



(f) SU

Figure 9. Count of logging questions for various programming languages for each of the six websites

sion provide an overview of most discussed logging topics. This insight will help developers to keep in mind these discussions while developing logging tools and libraries. It would also aid to keep a track of logging-related issues and needs of programmers. Additionally, it also provides insight about cross-discipline logging tools and practices, for example, the transaction log is used in both server environment (SF), Stack Overflow (SO) and database (DB) while event-logging practice is observed in super-user (SU) and Stack Overflow (SO).

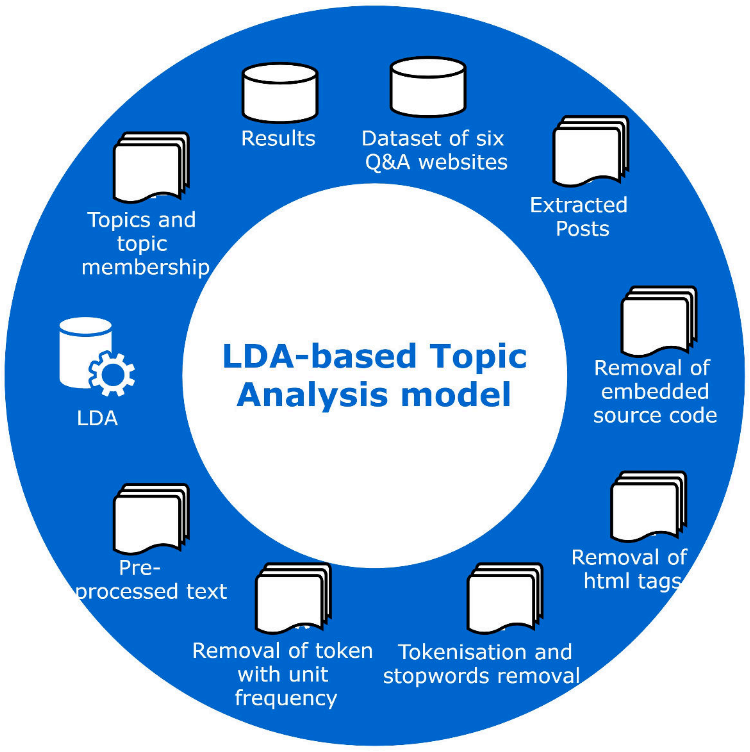

LDA-based Topic Analysis model
Dataset of six Q&A websites
Extracted Posts
Removal of embedded source code
Removal of html tags
Tokenisation and stopwords removal
Removal of token with unit frequency
Pre-processed text
LDA
Topics and topic membership
Results

Figure 10. Pre-processing steps of LDA

3.8.1. RQ6: What are the main discussion logging topics in various websites?

Motivation: In this RQ, we analyze main *discussion topics* present in the logging questions. Identifying major logging discussion topics can be beneficial in finding logging tools and libraries in which software developers face most difficulties. Since, we are analyzing logging topics on six websites, it can also be beneficial in identifying types of problems that developers face on each website.

Approach: We use LDA algorithm for identifying topics present in the logging questions in each website. We use Python library Gensim [34] in our work. LDA algorithm require three input parameters: *corpus*, *number of topics*, and *number of iterations*. Hence we first create our corpus. We extract *title* and *description* of each question to build the initial corpus. We apply five pre-processing steps to clean the initial corpus (refer to Figure 10 for details). First, we remove all the source code snippets from the description of the questions, i.e. we remove all the content between '<code>' and '<code>' tag. Second, we remove all the HTML tags such as '<p>, <a herf... >' from the description of the questions. Third, we remove all the English stop words, i.e. 'the', 'is', etc. Stop words are non-content bearing terms that do not add meaningful insight towards our goal. Fourth, we apply stemming to convert words to their root form. For example, the term 'programmers' is converted to 'programmer'. We use Porter Stemmer algorithm for stemming [35]. Fifth, we remove all the words that occur only once in the corpus. LDA takes *number of topics, i.e. K*, as the second parameter. Since, there is no best value known for K that is suitable for all types of dataset. We vary the value of K from 10 to 50 and select the value giving the best results. The third parameter to LDA algorithm in the number of iterations. In this work, we set the number of iterations equal to 500.

Results: We analyze results given by the LDA algorithm. The results showed that for all the websites we were getting meaningful topics for $K = 50$, hence, we selected this value. Table 8 shows some of the topics discovered for each website. Following is the detailed discussion about the topics obtained in each website:

SO: We discovered logging topics related to different domains on SO website. For example, logging topics in OOP, classes, etc. Logging topics in different programming languages like Java, Python, C#, C. We also find topics related to programming features like object oriented programming language, file handling, class features. Table 10 shows an illustrative example of a logging question for *Python* programming language asked on the SO website.

SF: In SF website, we observe logging topics related to issues to various servers like email server, tomcat server, Apache server. In addition, we observe topics related to networking and syslog. Table 10 shows an illustrative example of a logging question related to *networks* asked on the SF website.

SU: In SU website, we find logging topics related to Windows, USB, message server, command line, etc. Table 10 shows an illustrative example of a logging question related to *USB* asked on the SU website.

Table 8. Popular logging topic identified in all the six websites

Android Enthusiast, $K = 50$	
Topics	Words
document processing in system	document, write, store, readonly, writing, system
log in Android applications	adb, logcat, applications, Android, device, bootloader
log in tablet	adb, tablet, file, files, log, applications
log in Android phone	log, phone, app, Android, apps, time
log in device	log, logs, apps, storage, device, problem
apps	apps, google, exception, phone, install, exit
log in network connection	log, WiFi, network, IP, DHCP, connect, logcat
Database Administrators, $K = 50$	
Topics	Words
Backup	Restore Backup, log, transaction, backups, restore, recovery
Binarylog in MySQL	Binlog, MySQL, Binlogs, Binary, Logs
databases	null, bigint, int, varchar15, commit, set
rebuliding indexes	index, rebuild, progress, task, query, source, end
SQL features	table, rows, column, insert, values, id
memory allocation	reserved, allocated, commited, pages, kb, node
transaction-log in databases	select, query, transaction, log, table, join
log in databases server	logs, db2, server2, server, databases, transaction
binlog in MySQL server	binlog, mysqlbinlog, query, MySQL, server, endlogpos
PostgreSQL	master, slave, Postgres, PostgreSQL, archive, wal
log in SQL databases server	log, SQL, database, databases, server, mirroring
log in Oracle databases	redo, Oracle, database, log, logs, files
Server Fault, $K = 50$	
Topics	Words
log in network	TCP, UDP, log, port, lo, accept
syslog in messaging	syslogng, log, destination, source, get, messages
log in email server	ip, email, mail, Logwatch, server, postfix
log in Tomcat server	log, file, Tomcat, logs, server, files
log in Apache server	Apache, log, errorlog, logs, acesslog, server
different syslog in messages	rsyslog, syslog,varlogmessages, log, logging, logs
log in SQL server	server, log, database, server, backup, transaction

Table 8 continued

Stack Overflow, $K = 50$	
Topics	Words
OOP	int, include, class, void, char, const
features of class	public, void, static, class, private, new, import, null, return, string
log in Python	logging, logger, import, Python, log, logback
log in Java	log4j, spring, maven, class, log4j2, log4jwarn, logger
file handling	file, included, main, line, appender, stdout
programming features	undefined, reference, const, function, stdallocator, external
log in files	file, log, files, logs, line, write
log in Java Tomcat	file, log, log4j, logger, Tomcat, log4jproperties
C programing	include, const, return, int, typedef, function
log in Python file	logging, logger, file, log, Python, logback
log in file using C#	log4net, appender, file, log, using, config
Superuser, $K = 50$	
Topics	Words
log in Windows computer	Windows, log, application, logs, computer, service
log using USB	USB, device, plugged, logs, file, drive
syslog in message server	syslog, server, log, syslogd, mesaages, information
log in opengl file	system, opengl, log, logs, file, extension
log in file using command line	file, log, command, log, commands, output
Software Engineering, $K = 50$	
Topics	Words
Java	public, void, static, try, catch, throw, class, exception
log in client server	log, client, server, clients, logger,request
log in OOP's	exception, log, throw, method, catch, logging
software	software, license, disclaimer, copyright, warranty, warranties
log in databases	log, database, table, SQL, file, user
log in user application	log, user, application, logging, logged, data
log and exceptions	logger, log, exceptions, exception, logging, logginggetlogger
logging in project	project, logging, compiled, library, shared, core
log in files	log, files, appender, file, tests, application
multithreading in C in Unix	multithreading, Unix, C, code, library, boost
log in applications	logs, log, message, loglevel, application, information
log in systems	system, logging, libraries, log, developer, exception
logging in languages and OS	log4cxx, log4j, C, Java, Windows, Linux

Table 9. Categorical detail of observed logging-related tags

Category	Name	Tags	Websites	Description
General	Logging	logging	AE, DB, SE, SU, SO	Consider base of our analysis. Logging tag is used in all the six websites.

Table 9 continued

Category	Name	Tags	Websites	Description
General	Log-shipping	log-shipping	DB, SF, SO	It is a process of restoring transaction-log files on a standby server after creating a backup of transaction-logs on a primary database
	Log Files	logfiles, log-files, transaction-log	SO, SF, SU	It is a record of events that occurs in an operating system, software runs, etc. Transaction Log and Event logs forms a sub category of log files [36].
Types of Logging	Transaction-log	transaction-log	DB, SO, SF	A transaction log is a log of communication or transactions between a system and clients of that system.
	Event Log	event-log	SO, SU	Event logs aims to provide an audit trail that can be employed to understand the activity of the system and to diagnose problems. They forms the basis of understand activities of complex systems such as server applications.
	Error-log	error-log, error-logging	SO, DB, SU	Error-log is the collection of errors encountered during execution of a program.
	Binary-log	binlog, binarylog	SF, SO, DB	A binary log consist of binary log files and an index and is similar to transaction-log. They are used to restore data after backup.
Syslog and Syslog utilities	Syslog	syslog	SO, SF, SU	Syslog is the standard protocol for message logging.
	Rsyslog	ryslog	SF, SO	It is widely used in Unix and Unix-like operating system as a utility for transferring log messages in an IP network. It extends basic syslog protocol to content-based filtering along with providing features such as using TCP for transport.
	syslogd	syslogd	SF, SU	It provides provision for system logging and kernel message trapping. It supports both local and remote logging.
	syslog-ng	syslog-ng	SO, SF	syslog-ng extends syslogd model by adding content-based filtering, flexible configuration, TCP transport, etc.
Logging Tools and Libraries	Graylog	graylog, graylog2	SO, SF	Graylog is an Open-Source log capture tool and provides analysis solution for operational intelligence.
	NXLog	nxlog	SO, SF	NXLog is a log collector and supports log collection from multitude of sources and formats, e.g. event logs from TCP, UDP, file, databases, syslog, Windows event log, etc.
	Logparser	logparser	SO, SF, DB	Logparser is a command line tool designed to automate tests for Internet Information Services (IIS) logging.
	Logwatch	Logwatch	SF, SU	Logwatch is a log parser and analyser.
	Logstash	logsash	SO, SF	Logstash is used for managing events and logs. It deals with log processing, storage and searching.

Table 9 continued

Category	Name	Tags	Websites	Description
Logging Tools and Libraries	Hugo	hugo	SO, SU, SF	The Hugo logging plugin is used to log debug statements with the help of annotations.
	Log4j	log4k	SO, SF	Log4j is Java-based logging utility developed by Apache Software Foundation.
	Lynx	lynx	SU, SO	Lynx is the Android logging library.
	mysqlbinlog	mysqlbinlog	DB, SO	It is a utility used to process binary and relay logs.
	Boost	Boost	SO, SF, SE	Boost is a C++ based logging library.

DB: In DB website, we observe logging topics related to various domains like backup, SQL features, indexes, memory allocation. Additionally, we observe topics related to DB servers like Oracle and MySQL servers. Table 10 shows an illustrative example of a logging question related to *transaction-log* asked on the DB website.
AE: In AE website, we observe topics related to document processing, tablet, Android phone, and logs in network connection. Table 10 shows an illustrative example of a logging question related to *networking* asked on the AE website.
SE: In SE website, we obtain a wide variety of logging topics related to OOP, Java, files, databases, etc. Additionally, we observe topics related to exception and multi-threading. Table 10 shows an illustrative example of a logging question related to *object oriented programming* asked on the SE website.
RQ6 conclusions: For each website, we obtain logging topics related to different features such as programming language, transaction log, networking (Table 9).

3.8.2. RQ7: What is the distribution of logging-related tags across various Stack Exchange websites? And how persuasive is the commonality between these tags along various Stack Exchange websites?

Motivation: In this RQ, we analyse logging-related tags. Logging related tags mostly represent logging libraries, tools and technologies. This is a pressing need of our analysis to investigate distribution of these logging tags across various Q & A websites. This can be beneficial to find common logging libraries across six-websites as tags are the medium of classification on these websites. Distribution of these logging tags across various websites will provide us cues regarding common logging tools and libraries employed across various environments their trends. This may help the developers to design a common tool across different platforms capable of solving multiple problems.
Approach: In order to provide a compendious analysis, for this RQ, we have considered logging-related tags that are present in at-least two websites.
Results: Results of this analysis is divided into four categories, first, General Logging tags, second, Syslog and Syslog-based utilities, third, types of logging, and fourth, logging tools and libraries. Figure 11 depicts the observed results of our analysis. Center of the bubble depicts logging-related tags while its diameter corresponds to observed frequency. Following are the results of each category:
General Logging tags: This category consists of tags namely logging, log-files and log-shipping. The chief objective of log shipping is to ensure high availability of database by creating backup server that can replace production server quickly. It is used by both server engineers and database administrator on SF and DB respectively along with that this technique is further supports by well-known servers and databases namely Mi-

Table 10. Example post from various websites of the selected topic: WS: Website, QID: Question Id

WS	Topic	QID	Title	Body
SO	log in Python	41666158	log4j/logback pass logger level as a parameter	I want to do something which seems really straightforward: just pass a lot of logging commands (maybe all, but particularly WARN and ERROR levels) through a method in a simple utility class. I want to do this in particular so that during testing I can suppress the actual output to logging by mocking the method which does this call.
SF	log in network	508349	Rsyslog not logging from remote server	I am trying to set up a centralized log server. I have central server (A) receiving logs via a remote server (B) on port 514. I know it is receiving these. Here are a few entries from a tcp-dump on port 514... I have made sure to restart rsyslog every time I edit rsyslog.conf and I am running the start daemon with the -r and -t flags, even though they are deprecated in my current version. So why isn't anything coming in on port 514 being written to test.log?
SU	log using USB	849950	Logging when someone connects or removes a USB device to/from a Windows machine	I am currently trying to find a way to log all of the connections and disconnections of USB devices from all of the Windows machines on our network. This information needs to automatically be logged to a file on the machine, this file can then be read by nxlog and then get shipped to our centralised logging platform for processing. I was hoping that this information would be logged by Windows logs automatically, but I found that while some information about USB removable storage appears to get logged to Event Viewer, this is quite limited information and doesn't pick up when USB keyboards and mice are connected and disconnected...
DB	Transaction log	6996	How do I truncate the transaction log in a SQL Server 2008 database?	How do I truncate the transaction log in a SQL Server 2008 database? What are possible best ways? I tried this from a blog as follows: 1) From the setting database to simple recovery, shrinking the file and once again setting in full recovery, you are in fact losing your valuable log data and will be not able to restore point in time. Not only that, you will also not be able to use subsequent log files. 2) Shrinking database file or database adds fragmentation. There are a lot of things you can do. First, start taking proper log backup using the following command instead of truncating them and losing them frequently.
AE	log in network connection	85114	Does Android save a log of its own IP addresses?	I have a WiFi network to which I connect at work. The IP address has always been DHCP, but today the DHCP server is down. If I can figure out what IP address I had, I can set it statically (after checking to make sure another device hasn't already taken it, via ping from my desktop). Does Android have a log anywhere of the IP addresses it is leased? I have root and thus can look at any file on my phone.
SE	log in OOP's	255372	Logging exception in multi-tier application	I'm building a multi-tier enterprise application using Spring. I have different layers: Controller, Business and Provider. Within the application I've built a custom error handling mini-framework that is based on a single RuntimeException which has an error code to discriminate different kind of errors...

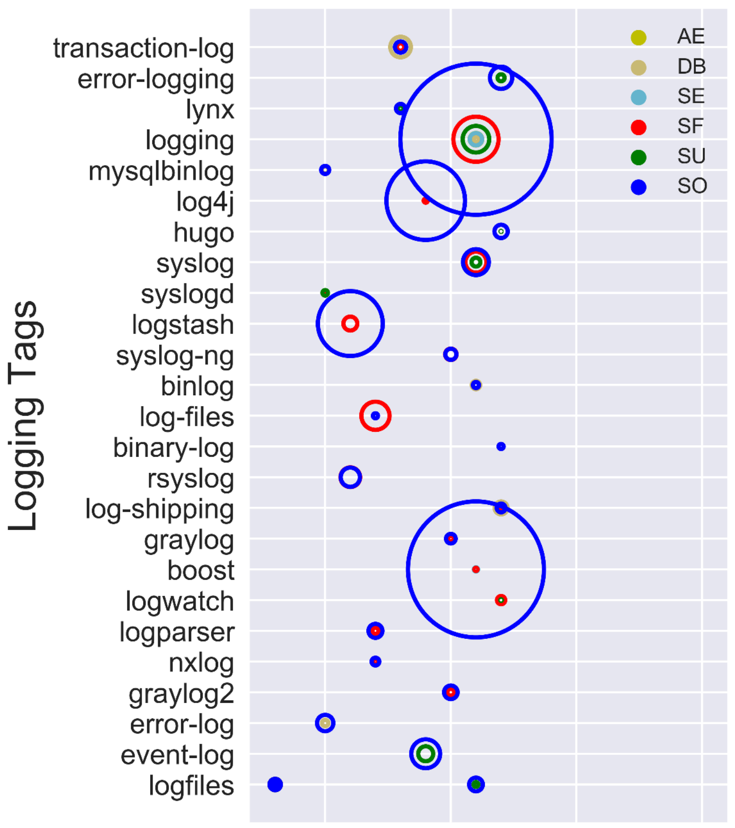


Figure 11. Bubble diagram of tags that are present in more than one website

crosoft SQL Server, 4D Server, PostgreSQL and MySQL [37–39].

Syslog and Syslog-based utilities: Syslog is the standard protocol for message logging. Syslog is commonly used for system management, security auditing and debugging analysis [40]. It provides provision for system logging and kernel message trapping. Many utilities extends syslog-based models, two of them are namely Rsyslog and Syslog-ng. These are commonly used by system programmers of SO and SF. Rsyslog website claims it to be Swiss army knife for logging [41]. Rsyslog extends basic syslog protocol to content-based filtering along with providing features such as using TCP for transport while Syslog-ng extends syslogd-model and thereby enhancing existing features [42, 43]. Both of these utilities are used for system logging while it is observed that Rsyslog is more popular on SO and SF than syslogd depending upon their tag frequency.

Types of Logging: Error Logging, Transaction logging, Event-logging and Binary-logging are some of the popular types of logging used by programmers of SO, SF, DB and SU. Error-logs are widely used for troubleshooting and bug fix [44]. Error logging is observed on SO, DB and SU. Event logs aims to provide an audit trail that can be employed to understand the activity of the system and to diagnose problems. They forms the basis of understand activities of complex systems such as server applications. Event logging is observed in SU and SO Highest number of transaction-log tags and binary log tags are observed in DB. This may be attributed to the fact that in order to allow the database to recover from crashes or other errors and to basically maintain consistent state, most of the databases maintain a transaction log [45]. Binary log is similar to transaction log, it records all changes in the databases including both data and structure [46].

Logging Tools and Libraries: Many logging tools and libraries are used by programmers to manage and process logs. Some of the popular logging tools detected in our analysis are Graylog, Graylog2, NXLog, Logparser, Logwatch, Logstash, Hugo, Lynx, Log4j and mysqlbinlog. Graylog and NXLog tag is frequently observed in both SF and SO. Usage of Graylog may be attributed to the fact that Graylog is a server that collects log messages along with that provides an interface for analysis and monitoring [47]. This tools is frequently used by Server Administrators while NXLog's high-performance I/O layer make it capable of handling thousands of parallel client connections in order to process huge log volumes [48]. Thus making it suitable for use in Server Environments. Logwatch provides feature to deliver a unified report of all activity on a server through command line or email to the administrator [49, 50]. Logparser is designed to automate tests for IIS logging. IIS is an extensible web server created by Microsoft used by database and server administrators. It aims to provide query axis to text-based data for, e.g. log files, XML, CSV, etc. Logparser is frequently observed on SF and SO while Logwatch is chiefly present on SF only. mysqlbinlog is used for processing binary log files and usage of this tag is observed in SO and DB [51]. Among all the above mention tools, frequency of usage of Logstash tag is maximum on SF while frequency of Log4j is maximum on SO. Logstash is one of the components of ELT-stack. This ELT-stack combination is widely used by Wikimedia Foundation. Logstash collects all the log-events sent by Wikimedia applications and stores them in an Elasticsearch cluster followed by use of font-end client Kibana in order to filter and display messages [52].

RQ7 conclusions: Syslog-based model Rsyslog is more popular on SO and SF than Syslogd-based model syslogng. System programmers tends to use transaction and binary logging more than error and event logging. This may be attributed to the fact that error and event logging is not observed in SF while transaction and binary logging is observed. DB Administrators tends to use transaction logging and binary logging frequently in order to maintain consistency of their database. Among all the websites observed tag frequency of transaction and binary logging is highest for DB. Among all the logging tools and libraries namely Logparser, Logwatch, NXLog, GrayLog, Logstash, Hugo, Log4j, the popularity of Logstash is maximum on SF while popularity of Log4j is maximum on SO in terms of tag-frequency.

4. Threats to validity

In this section, we discuss various threats to validity related to the results presented in this work.
Threats to external validity: In our study, we conduct experiments on 82 K logging questions from six different Q & A websites of the stack exchange network. These websites are subject to a general audience (e.g. SO) and specific audiences (e.g. AE, SF, DB, SU, and SE). Hence, we have depicted the results of source code logging analysis separately for each domain. Logging-related results can be generalized within a domain but may not be generalized across domains. However, results of SO provides a basic level of generalization considering its vast audience across multiple domains.
Threats to internal validity: For topic generation using LDA, we have used $K = 50$ for all six websites. However, this is done irrespective of the size of the corpus of each website. Further, all the code snippets were removed from the analysis using regular expressions along with HTML tags. We notice that previous studies analyzing content from Stack Exchange websites have also removed source code present in the description of posts [16]. We further minimize the threats of internal validity by using built-in Python libraries (for example, Sklearn, NLTK) for doing data processing.
Threats to construct validity: It is concerned with the identification of logging-related tags that formed the basis of our study and further interpretation of topics. Threats to construct validity is categorized into 3 main

parts: **the construction of programming language-related tags**, **construction of general logging-related tags** and **Interpretation of LDA-topics**. First, There are several programming languages. However, we selected 6 programming language through ostensible randomization. Additionally, we use term programming languages for programming languages as well as scripting languages (JS). In order to determine the number of logging-related questions corresponding to programming languages, we have used various logging libraries, tools, APIs, etc. specific to that programming languages. Table 7 depicts these logging-related libraries, tools, APIs, etc. corresponding to six-programming languages. These libraries are selected after rigorous manual exploration of the internet and existing research by best of our knowledge but due to several programming-language related libraries in the field, there may exist some libraries left unexplored. Hence, C++ and Java have more logging-related questions than Python and C may be because some of Python or C related logging libraries could be left unexplored. Moreover, Boost is a set of libraries written in C++ and aims to provide support for a multitude of tasks, for example, algebra, unit testing, multithreading, etc. Thus, Boost tag consists of a set of logging as well as non-logging libraries for C++ which can affect the results of the actual number of logging questions concerning C++. Second, Logging-related Tags used in our analysis are depicted in Table 7. Mostly, these tags are the comprehensive collection of logging-related terms (transaction-log, log-files, etc.) and logging-related libraries, Tools, and APIs (SL4J, Logstash, etc.). Some of these tags are used by developers in more than one context, for example, Observed results of Lynx and Hugo can be inconsistent as Lynx is a logging library as well as a text-mode web browser while Hugo is also a logging library but also a static site generator written in Go. Third, Interpretation of topics generated from LDA is not an easy task [53] and can be subjective. Thus, first, second author and corresponding author understood the topics and derived the topic labels and other authors verified them. In the cases where topics were hard to interpret, we further studied the questions related to them in order to drive a label.

5. Conclusions and future work

Logging is an important software development practice. Log statements present in the source code are used to record important runtime information. Software practitioners can use this information at the time of debugging. In past, several research studies have been conducted that propose solutions to help software developers in source code logging. These solutions are helpful but at present there is no study that analyzes the issues that software developers face while logging. In this paper, we perform a three dimensional, empirical study of logging questions asked on the six popular Q & A websites. We perform statistical, programming language and content analysis of logging questions. Our analysis helped us to gain insight about the logging discussion happening in six different domains of the Stack Exchange websites.

Our analysis provides an insight about the logging needs of software developers. Results of our in-depth empirical study show that logging questions are pervasive in all the Q & A websites. The mean time to get accepted answer for logging questions on SU and SF websites are much higher as compared to other websites. It also shows that a large number of logging question invite a great amount of discussion in the SoftwareEngineering Q & A website. We have found that software developers face most of the logging issues in C++ and Java. It shows that the trend of number of logging questions is increasing for Java, Python, and JavaScript, whereas, it is decreasing or constant for C, C++, C#. Researchers can use these results to fine tune the automated logging tools proposed by them. Companies can use these results to fine-tune their tools and to decide which technique to support.

Our analysis also shows that different websites have different dominant programming language. For the SO website C++ and Java are dominant language whereas for the SF and SU website, 'C' is the dominant programming language. Researchers can use this information, for example, if they are providing automated logging tool for server they can target it with 'C' language, whereas, if they are making general purpose logging tool, they can target it with 'C++' or 'Java'.

Since, this is the first study of on logging issues of Q & A websites, in this we explored different dimensions of logging question in future, we will explore more specific logging problems faced by software practitioners. We plan to extend this work in several dimensions. First, at present we have performed topic analysis using question title and description only. In future, we plan to perform topic analysis of *answers* as well. Second, we plan to perform sentiment analysis of comments associated with logging questions. To find the overall sentiment of users about logging. Third, we plan to perform topic analysis for small time intervals like 1–3 months in order to to find how topic related to logging are changing over the period of time. Fourth, we will perform analysis of most popular logging questions irrespective of the website on which they are asked.

6. Acknowledgment

We thank our advisor and mentor Dr. Ashish Sureka for providing guidance and inputs in this project.

References

[1] Q. Fu, J.G. Lou, Y. Wang, and J. Li, "Execution anomaly detection in distributed systems through unstructured log analysis," in *Proceedings of the Ninth IEEE International Conference on Data Mining*, ICDM '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 149–158.

[2] K. Nagaraj, C. Killian, and J. Neville, "Structured comparative analysis of systems logs to diagnose performance problems," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, 2012, pp. 26–26.

[3] S. Lal and A. Sureka, "LogOpt: Static feature extraction from source code for automated catch block logging prediction," in *Proceedings of the 9th India Software Engineering Conference (ISEC)*, 2016, pp. 151–155.

[4] S. Lal, N. Sardana, and A. Sureka, "LogOptPlus: Learning to optimize logging in catch and if programming constructs," in *Proceedings of the IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 1, June 2016, pp. 215–220.

[5] H. Li, W. Shang, and A.E. Hassan, "Which log level should developers choose for a new logging statement?" *Empirical Software Engineering*, Vol. 22, No. 4, 2017, pp. 1684–1716.

[6] S. Kabinna, C.P. Bezemer, W. Shang, and A.E. Hassan, "Logging library migrations: A case study for the Apache Software Foundation projects," in *Proceedings of the 13th International Conference on Mining Software Repositories*, MSR '16. New York, NY, USA: ACM, 2016, pp. 154–164.

[7] S. Lal, N. Sardana, and A. Sureka, "Improving logging prediction on imbalanced datasets: A case study on open source java projects," *International Journal of Open Source Software and Processes (IJOSSP)*, Vol. 7, No. 2, 2016, pp. 43–71.

[8] StackExchange Community, *StackOverflow home page.* [Online]. https://stackoverflow.com/ [accessed: 26.12.2017].

[9] StackExchange Community, *Serverfualt stack exchange home.* [Online]. https://serverfault.com/ [accessed: 26.12.2017].

[10] StackExchange Community, *Superuser Stack Exchange home page.* [Online]. https://superuser.com/ [accessed: 26.12.2017].

[11] StackExchange Community, *Database Administrators Stack Exchange home page.* [Online]. https://dba.stackexchange.com/ [accessed: 26.12.2017].

[12] StackExchange Community, *Android Enthusiasts home page.* [Online]. https://android.stackexchange.com/ [accessed: 26.12.2017].

[13] StackExchange Community, *SoftwareEngineering home page.* [Online]. https://softwareengineering.stackexchange.com/ [accessed: 26.12.2017].

[14] G. Pinto, F. Castor, and Y.D. Liu, "Mining questions about software energy consumption," in *Proceedings of the 11th Working Conference on Mining Software Repositories.* ACM, 2014, pp. 22–31.

[15] M. Linares-Vásquez, B. Dit, and D. Poshyvanyk, "An exploratory analysis of mobile development

issues using Stack Overflow," in *Proceedings of the 10th Working Conference on Mining Software Repositories*. IEEE Press, 2013, pp. 93–96.

[16] A. Barua, S.W. Thomas, and A.E. Hassan, "What are developers talking about? an analysis of topics and trends in Stack Overflow," *Empirical Software Engineering*, Vol. 19, No. 3, 2014, pp. 619–654.

[17] B. Chen and Z.M.J. Jiang, "Characterizing logging practices in Java-based open source software projects – A replication study in Apache Software Foundation," *Empirical Software Engineering*, Vol. 22, No. 1, 2017, pp. 330–374.

[18] Q. Fu, J. Zhu, W. Hu, J.G. Lou, R. Ding, Q. Lin, D. Zhang, and T. Xie, "Where do developers log? An empirical study on logging practices in industry," in *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion, 2014, pp. 24–33.

[19] S. Lal, N. Sardana, and A. Sureka, "Two level empirical study of logging statements in open source Java projects," *International Journal of Open Source Software and Processes (IJOSSP)*, Vol. 6, No. 1, 2015, pp. 49–73.

[20] W. Shang, M. Nagappan, and A.E. Hassan, "Studying the relationship between logging characteristics and the code quality of platform software," *Empirical Software Engineering*, Vol. 20, No. 1, 2015, pp. 1–27.

[21] D. Yuan, S. Park, and Y. Zhou, "Characterizing logging practices in open-source software," in *Proceedings of the 34th International Conference on Software Engineering*, (ICSE), 2012, pp. 102–112.

[22] H. Li, W. Shang, Y. Zou, and A.E. Hassan, "Towards just-in-time suggestions for log changes," *Empirical Software Engineering*, Vol. 22, No. 4, 2017, pp. 1831–1865.

[23] D. Yuan, S. Park, P. Huang, Y. Liu, M.M. Lee, X. Tang, Y. Zhou, and S. Savage, "Be conservative: Enhancing failure diagnosis with proactive logging," in *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation (OSDI)*, 2012, pp. 293–306. [Online]. http://dl.acm.org/citation.cfm?id=2387880.2387909

[24] J. Zhu, P. He, Q. Fu, H. Zhang, M. Lyu, and D. Zhang, "Learning to log: Helping developers make informed logging decisions," in *Proceedings of the IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE)*, Vol. 1, May 2015, pp. 415–425.

[25] S. Kabinna, C.P. Bezemer, W. Shang, and A.E. Hassan, "Examining the stability of logging statements," in *Proceedings of the 23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, 2016, pp. 326–337.

[26] S. Lal, N. Sardana, and A. Sureka, "ECLogger: Cross-project catch-block logging prediction using ensemble of classifiers," *e-Informatica Software Engineering Journal*, Vol. 11, No. 1, 2017, pp. 9–40.

[27] S. Beyer and M. Pinzger, "A manual categorization of android app development issues on Stack Overflow," in *Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on*. IEEE, 2014, pp. 531–535.

[28] X.L. Yang, D. Lo, X. Xia, Z.Y. Wan, and J.L. Sun, "What security questions do developers ask? A large-scale study of Stack Overflow posts," *Journal of Computer Science and Technology*, Vol. 31, No. 5, 2016, pp. 910–924.

[29] H. Malik, P. Zhao, and M. Godfrey, "Going green: An exploratory analysis of energy-related questions," in *Proceedings of the 12th Working Conference on Mining Software Repositories*. IEEE Press, 2015, pp. 418–421.

[30] C. Nagy and A. Cleve, "Mining Stack Overflow for discovering error patterns in SQL queries," in *Software Maintenance and Evolution (ICSME)*. IEEE, 2015, pp. 516–520.

[31] StackExchange Community, *StackExchange*. [Online]. https://stackexchange.com/ [accessed: 26.12.2017].

[32] Quora Community, *Quora Home Page*. [Online]. https://www.quora.com/ [accessed: 26.12.2017].

[33] StackExchange Community, *What does it mean when an answer is "accepted"*. [Online]. https://stackoverflow.com/help/accepted-answer [accessed: 26.12.2017].

[34] Python Community, *Latent Dirichlet Allocation (LDA) in Python*. [Online]. https://radimrehurek.com/gensim/models/ldamodel.html [accessed: 9.04.2018].

[35] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.

[36] Neurobs, *Neurobs*. [Online]. https://www.neurobs.com/pres_docs/html/03_presentation/07_data_reporting/01_logfiles/index.html [accessed: 12.03.2018].

[37] Wikipedia, *4th Dimension (software).* [Online]. https://en.wikipedia.org/wiki/4th_Dimension_(software) [accessed: 12.03.2018].

[38] PostgreSQL, *Warm Standby Servers for High Availability.* [Online]. http://www.postgresql.org/docs/8.2/static/warm-standby.html [accessed: 12.03.2018].

[39] MySQL Community, *Reference Manual on Configuring Replication.* [Online]. https://dev.mysql.com/doc/refman/5.7/en/replication-configuration.html [accessed: 12.03.2018].

[40] Network Working Group, *The Syslog Protocol.* [Online]. https://tools.ietf.org/html/rfc5424 [accessed: 12.03.2018].

[41] Rsyslog Community, *Rsyslog.* [Online]. https://www.rsyslog.com/ [accessed: 12.03.2018].

[42] Syslog-ng Community, *Reliable, scalable, secure central log management.* [Online]. https://syslog-ng.com/ [accessed: 12.03.2018].

[43] Python Community, *syslogd – Linux man page.* [Online]. https://linux.die.net/man/8/syslogd [accessed: 12.03.2018].

[44] Techopedia, *Error Log.* [Online]. https://www.techopedia.com/definition/26306/error-log [accessed: 12.03.2018].

[45] T.A. Peters, "The history and development of transaction log analysis," *Library Hi Tech*, Vol. 11, No. 2, 1993, pp. 41–66.

[46] MariaDB Community, *Binary Log.* [Online]. https://mariadb.com/kb/en/library/binary-log/ [accessed: 12.03.2018].

[47] StackOverflow Community, *Graylog.* [Online]. https://stackoverflow.com/tags/graylog/info [accessed: 3.05.2018].

[48] StackOverflow Community, *NXLOG.* [Online]. https://stackoverflow.com/tags/nxlog/info [accessed: 3.05.2018].

[49] archlinux, *Logwatch.* [Online]. https://wiki.archlinux.org/index.php/Logwatch [accessed: 3.05.2018].

[50] Bjorn, F. Crawford, J. Pyeron, J. Soref, K. Bauer, M. Tremaine, O. Poplawski, and S. Jakobs, *Logwatch.* [Online]. https://sourceforge.net/p/logwatch/wiki/Home/ [accessed: 12.03.2018].

[51] MySQL Community, *mysqlbinlog – Utliity for Processing Binary Log Files.* [Online]. https://logging.apache.org/log4j/2.x/ [accessed: 12.03.2018].

[52] Wikitech, *Logstash – Wikitech.* [Online]. https://wikitech.wikimedia.org/wiki/Logstash [accessed: 12.03.2018].

[53] A. Hindle, C. Bird, T. Zimmermann, and N. Nagappan, "Do topics make sense to managers and developers?" *Empirical Software Engineering*, Vol. 20, No. 2, 2015, pp. 479–515.

e-Informatica Software Engineering Journal, Volume 13, Issue 1, 2019, pages: 141–202, DOI 10.5277/e-Inf190105

Empirical Studies on Software Product Maintainability Prediction: A Systematic Mapping and Review

Sara Elmidaoui*, Laila Cheikhi*, Ali Idri*, Alain Abran**
*SPM Team, ENSIAS, Mohammed V University in Rabat, Morocco
**Department of Software Engineering and Information Technology, École de Technologie Supérieure, Montréal, Canada
sara.elmidaoui@um5s.net.ma, laila.cheikhi@um5.ac.ma, ali.idri@um5.ac.ma, alain.abran@etsmtl.ca

Abstract

Background: Software product maintainability prediction (SPMP) is an important task to control software maintenance activity, and many SPMP techniques for improving software maintainability have been proposed. In this study, we performed a systematic mapping and review on SPMP studies to analyze and summarize the empirical evidence on the prediction accuracy of SPMP techniques in current research.
Objective: The objective of this study is twofold: (1) to classify SPMP studies reported in the literature using the following criteria: publication year, publication source, research type, empirical approach, software application type, datasets, independent variables used as predictors, dependent variables (e.g. how maintainability is expressed in terms of the variable to be predicted), tools used to gather the predictors, the successful predictors and SPMP techniques, (2) to analyze these studies from three perspectives: prediction accuracy, techniques reported to be superior in comparative studies and accuracy comparison of these techniques.
Methodology: We performed a systematic mapping and review of the SPMP empirical studies published from 2000 up to 2018 based on an automated search of nine electronic databases.
Results: We identified 82 primary studies and classified them according to the above criteria. The mapping study revealed that most studies were solution proposals using a history-based empirical evaluation approach, the datasets most used were historical using object-oriented software applications, maintainability in terms of the independent variable to be predicted was most frequently expressed in terms of the number of changes made to the source code, maintainability predictors most used were those provided by Chidamber and Kemerer (C&K), Li and Henry (L&H) and source code size measures, while the most used techniques were ML techniques, in particular artificial neural networks. Detailed analysis revealed that fuzzy & neuro fuzzy (FNF), artificial neural network (ANN) showed good prediction for the change topic, while multilayer perceptron (MLP), support vector machine (SVM), and group method of data handling (GMDH) techniques presented greater accuracy prediction in comparative studies. Based on our findings SPMP is still limited. Developing more accurate techniques may facilitate their use in industry and well-formed, generalizable results be obtained. We also provide guidelines for improving the maintainability of software.

Keywords: systematic mapping study, systematic literature review, software product maintainability, empirical studies

Submitted: 27 June 2018; Revised: 5 April 2019; Accepted: 6 April 2019; Available online: 17 July 2019

1. Introduction

Maintainability of a software product is defined in SWEBOK [1] as a quality characteristic that "must be specified, reviewed, and controlled during the software development activities in order to reduce maintenance costs". Many techniques for software product maintainability prediction (SPMP) have been proposed as a means to better manage maintenance resources through a defensive design [2]. However, predicting software maintainability remains an open research area since the maintenance behaviors of software systems are complex and difficult to predict [3]. Moreover, industry continues to search for appropriate ways to help organizations achieve reliable prediction of software product maintainability.

A number of studies have been conducted in this context [4–9]. For instance, Riaz et al. [4] conducted a systematic literature review (SLR) on a set of 15 primary studies dating from 1985 to 2008 to investigate techniques and methods used to predict software maintainability. They found that the number of studies varied from one to two per year illustrating that this research topic was still in emergence in 2008 and had not yet reached a certain level of maturity. Moreover, they showed that the choice among prediction models for maintainability was not obvious (12 out of 15 studies had proposed models). Size, complexity and coupling were commonly used independent variables for maintainability, while maintainability expressed in terms of an ordinal scale based on expert judgment was the most commonly used dependent variable. A subsequent SLR (from 1985 to 2010) by Riaz [5] identified seven primary studies that focused on relational database-driven applications (RDBAs). The results showed little evidence for maintainability prediction for relational database-driven applications. He found that: expert judgment was the most common prediction technique, coupling related measures were the most common predictors, and subjective assessment was the most common dependent variable.

Orenyi et al. [6] conducted a survey on object-oriented (OO) software maintainability using a set of 36 studies published between 2003 and 2012. The authors investigated the use of a quality model, sub-characteristics or measures and techniques, and noted that regression analysis techniques were the most used (31% of the 36 studies). Dubey et al. [7] provided an overview of a set of 21 studies on maintainability techniques for OO systems published between 1993 and 2011. In these latter two studies (not SLRs) the authors did not provide a detailed analysis. Fernandez-Saez et al. [8] conducted a systematic mapping study (SMS) on a set of 38 primary studies (collected from 1997 to 2010) in order to discover empirical evidence related to the use of UML diagrams in source-code maintenance and the maintenance of UML diagrams themselves. They found that "the use of UML is beneficial for source code maintenance, since the quality of the modifications is greater when UML diagrams are available, and most research concerns the maintainability and comprehensibility of the UML diagrams themselves". To explore the use of UML documentation in software maintenance, the authors have published results from a survey of software industry maintenance projects [9] by 178 professionals from 12 different countries. The findings were summarized as follows: "59% indicated the use of a graphical notation and 43% UML, most effective UML diagrams for software maintenance were class, use case, sequence and activity diagrams, the benefits of using UML diagrams result in less time needed for a better understanding and, thus an improved defect detection, and larger teams seem to use UML more frequently in software maintenance".

A summarized context of this related work is presented in Table 1 in terms of: purpose of the study, research or mapping questions addressed, type of study (SLR, SMS or another form of literature review, such as survey, review, etc.), period of collection, and the number of primary studies for each study.

As can be seen from Table 1, while all studies shared an interest in the maintainability of the software, they focused on different aspects or topics within the field. The period of collection and number of primary studies varied among the reviews. Only three studies conducted a rigorous review with SLR and SMS addressing

Table 1. Summarized context of related work

Study ID	Purpose	Research or mapping questions addressed	Type	Period of collection	#of studies
[4]	Understand the state of the art of the software maintainability prediction techniques and metrics.	1) techniques, 2) accuracy measures, 3) independent variables, 4) dependent variables.	SLR	1985–2008	15
[5]	Understand the state of the art of the software maintainability prediction techniques and metrics in RDBAs.	1) techniques, 2) accuracy measures, 3) independent variables, 4) dependent variables.	SLR	1985–2010	7
[6]	Review existing studies in the area of OO software maintainability measurement.	Not provided	Survey	2003–2012	36
[7]	Review of studies on software maintainability model with OO system.	Not provided	Survey	1993–2011	23
[8]	Review of studies on maintenance of UML diagrams and their use in the maintenance of code.	1) UML Diagrams, 2) dependent variable, 3) state of the art, 4) factors	SMS	1997–2010	38
[9]	Survey on the use of UML in software maintenance in order to gather information and opinions from a large population.	Not provided	Survey	February to April of 2013	–

some research or mapping questions. The SMS [8] focused on empirical studies concerning the maintenance of UML diagrams and their use in the maintenance of code. However, the scope of this study was broader and focused not only on UML diagrams but also provided a state-of-the art review of software product maintainability prediction in general. The SLR [4] addressed four research questions (see Table 1), while our study addressed additional questions related to publication trends, publication sources, research types, empirical approaches, software application types, datasets, and tools used to gather these independents variables. Moreover, in our study, to provide answers to the mapping questions, we classified the selected studies according to a set of proposed criteria, whereas study [4] only extracted data for some research questions, presenting them in tables as reported in the primary studies without providing any analysis. Furthermore, none of the previous studies dealt with the accuracy of SPMP techniques whereas our study analyzes and summarizes the evidence regarding prediction accuracy of SPMP techniques as well as identifies the most accurate in comparative studies.

Since the publication of SLRs [4, 5] and SMS [8] studies a number of new empirical studies have been published, some proposing new techniques, such as machine learning techniques, others evaluated existing ones, while still others provided comparative studies to identify the most accurate. Furthermore, since the first SLR on software maintainability was published in 2008, it was important to investigate what further research had occurred since. Moreover, the number of primary studies investigated was very small (from 7 to 15) and the results obtained cannot be conclusive. To establish the state-of-the-art on this topic and reach a certain level of external

Table 2. Mapping and research questions

ID	Mapping questions	Motivation
MQ1	How has the frequency of SPMP studies changed over time?	To identify the publication trend of SPMP studies over time.
MQ2	What are the main publication sources?	To identify what and how many publication sources for SPMP studies.
MQ3	What research types were used?	To identify the different research types used in SPMP studies.
MQ4	What empirical approaches were used?	To identify the empirical approaches that have been used to validate SPMP techniques.
MQ5	What types of software applications were used?	To identify the software application types on which the SPMP studies focused.
MQ6	What datasets were used?	To identify the datasets used for SPMP empirical studies, including the number of projects in the empirical studies.
MQ7	What dependent and independent variables were used?	To identify: A) How maintainability was expressed in terms of the variable to be predicted (e.g. dependent variable). B) What measures or factors were used as predictors (i.e. independent variables) for SPMP. C) Successful predictors for maintainability as reported by the selected studies. D) Tools used to gather predictors.
MQ8	What techniques were used in SPMP?	To identify and classify the techniques used in SPMP studies.
ID	Research questions	Motivation
RQ1	What is the overall prediction accuracy of SPMP techniques?	To identify to what extend the SPMP techniques provide accurate prediction.
RQ2	Which SPMP techniques were reported to be superior in comparative studies?	To identify SPMP techniques reported to be superior in comparative studies.
RQ3	Which of the SPMP techniques reported to be superior in comparative studies also provided greater accuracy?	To compare SPMP techniques that have been reported to be superior in the comparative studies using the same prediction context in terms of accuracy prediction.

validity [4], research published during the last 10 years of studies providing empirical validation of their finding needs to be investigated. This study differs from previous reviews in several ways: it provides an up-to-date state-of-the-art review of SPMP (from 2000 to 2018), the search was conducted on nine digital libraries, a set of 82 primary studies were selected, and classification criteria were proposed for purposes of detailed and precise analysis of the results. A set of eight mapping questions (MQs) were addressed related to: (1) publication year, (2) publication source, (3) research type, (4) empirical approach, (5) software application type, (6) datasets, (7) independent variables (e.g. factors used as predictors) and dependent variables (e.g. how maintainability is expressed in terms of the variable to be predicted), and (8) techniques used, as well as a set of three research questions (RQs) related to: (1) prediction accuracy, (2) techniques reported superior in comparative studies and (3) accuracy comparison of these techniques (see Table 2). Therefore, the objective of this study was twofold:

– to classify SPMP studies according to the proposed criteria (see Table 3), and,
– to analyze and summarize the empirical evidence of SPMP technique accuracy prediction in current research.

The rest of the paper is organized as follows. Section 2 presents the methodology used to conduct the study including the mapping and research questions to be addressed, the research strategy and selection of the primary studies.

Table 3. Classification criteria

Property	Categories
Research types	Solution proposal (SP), evaluation research (ER)
Empirical approaches	History-based evaluation (HbE), case study (CS), experiment or family of experiments (Ex)
Software application types	Object-oriented applications (OOA), procedure-oriented applications (POA), web-based applications (WbA), service-oriented applications (SOA), component-based applications (CbA), not identified (NI)
Datasets	Software engineering researchers (SER), open source software systems/projects (OSS), private software projects/systems (PSP) dependent variable change, expert opinion, maintainability index, maintainability level, maintainability time, others
Independent variables	Chidamber and Kemerer (C&K), Li and Henry (L&H), class diagram, source code size, McCabe complexity (McCabe), software quality attributes, Martin's measures, Halstead measures, Brito e Abreu and Carapuça (BA&C), factors, coding rule measures, quality model for object-oriented design (QMOOD) measures, maintainability index (MI), web-based application (WbA) measures, Jensen measures, effort measures, sequence diagram, Lorenz and Kidd (L&K) measures, fault measures, database measures
Techniques	Machine learning (ML), artificial neural network (ANN), fuzzy & neuro fuzzy (FNF), regression & decision trees (DT), case-based reasoning (CBR), Bayesian networks (BN), evolutionary algorithm (EA), support vector machine & regression (SVM/R), inductive rule based (IRB), ensemble methods (EM), clustering methods (CM); statistical: regression analysis (RA), probability density function (PD), Gaussian mixture model (GMM), discriminant analysis (DA), weighted functions (WF), stochastic model (SM)

Section 3 summarizes the results by providing answers to the mapping questions. Section 4 provides the results of the research questions. Section 5 presents the threats to validity of the work. Section 6 offers conclusions and possible future directions.

2. Research methodology

In this study, we used the guidelines of Petersen et al. [10] for conducting systematic reviews, which include planning, conducting and reporting. According to Kitchenham, "Systematic Mapping Studies (SMS) use the same basic methodology as SLRs but aim to identify and classify all research related to a broad software engineering topic rather than answering questions about the relative merits of competing technologies that conventional SLRs address" [11]. In the planning step, the review protocol was developed which describes the procedure for conducting the review. The steps of this protocol are summarized as follows: (1) establishment of a set of mapping and research questions to address the issues related to the review, (2) identification of the search strategy including identification of search terms, selection of sources to be searched, and the search process, (3) selection of the set of primary studies using inclusion and exclusion criteria, (4) mapping of publications by extracting data from each selected study, and (5) data synthesis by grouping the overall results in order to facilitate analysis and provide answers to the mapping and research questions. The protocol was established by holding frequent meetings between authors. A detailed description of each of these steps is provided in the following subsections.

2.1. Mapping and research questions

In addition to our primary motivation to provide and summarize evidence from published empirical studies on SPMP, according to our set of criteria, we identified eight mapping questions

(MQs) and three research questions (RQs) – see Table 2.

The MQs are related to the structuring of the SPMP research area with respect to the properties and categories described in Table 3. These categories are defined and explained in the Tables A1 and A2 in the Appendix.

2.2. Search strategy

The search strategy used to identify the primary studies included the following steps: identify the search terms, apply these search terms to electronic databases to retrieve candidate studies, use the search process to ensure that all relevant studies are identified.

2.2.1. Search terms

The search terms were identified based on the MQs and RQs by identifying keywords, synonyms and alternative spellings. The main search terms were: "maintainability", "empirical", "software", "prediction", and "technique". Table 4 provides the main search terms and their alternatives spellings. As can be seen from Table 4, for alternative terms related to maintainability we considered all the maintainability sub-characteristics proposed in the standard ISO 9126 and used in previous SLRs [4, 5].

The search terms were derived using the following series of steps [12]:

– Define the main search terms matching the mapping questions listed above.
– Identify synonyms and alternative spellings for the main terms.
– Use the Boolean OR to concatenate synonymous and alternative terms in order to retrieve any record containing either (or all) of the terms.
– Use the Boolean AND to connect the main terms in order to retrieve any record containing all the terms

The following set of search terms were used to extract the primary studies: "(maintainability **OR** analyzability **OR** modifiability **OR** testability **OR** compliance **OR** stability) **AND** (empirical* **OR** evaluation* **OR** validation* **OR** experiment* **OR** control* experiment **OR** case study **OR** survey) **AND** (software product **OR** software **OR** application **OR** system **OR** software engineering) **AND** (predict* **OR** evaluat* **OR** assess* **OR** estimat* **OR** measur*) **AND** (method* **OR** technique* **OR** model* **OR** tool* **OR** approach*)"

2.2.2. Literature resources

To search for primary studies, nine relevant and important digital libraries in software engineering used in previous SLRs and SMSs [4, 5, 12] were chosen, which included journals, books, and conference proceedings from: IEEE Xplore, Science Direct, Springer Link, Ebsco, ACM Digital Library, Google Scholar, Scopus, Jstore, and DBLP. The preconstructed search terms established in the first step were applied to this set of nine digital libraries. The search focused on title, abstract and keywords, and ranged from 2000 to 2018.

2.2.3. Search process

To ensure selection of the maximum number of studies related to SPMP, a first round search (automated) was performed using the search terms on each digital library to gather the overall set of candidate studies. A second search round (manual) was performed, which consisted of examining the reference lists of the set of candidate studies in order to identify new candidates based on

Table 4. Search terms

Main terms	Alternative terms
maintainability	analyzability, modifiability, testability, stability, compliance
empirical	evaluation, validation, experiment, control experiment, case study, survey
software	software product, software, application, system, software engineering
prediction	prediction, evaluation, assessement, estimation, measurement
technique	method, technique, model, tool, approach

the title. If the full study was not available, the authors were contacted to obtain a copy of the published work.

2.3. Study selection

After applying the search process, the full text of the candidate studies retrieved were assessed by two authors according to the following inclusion and exclusion criteria.

- **Inclusion criteria (IC)**: (1) empirical studies addressing prediction or assessment of software product maintainability and/or its sub-characteristics, (2) empirical studies using SPMP techniques.
- **Exclusion criteria (EC)**: (1) studies that discuss the process of software maintenance, (2) studies that concentrate on software maintainability generally and do not present a technique to predict the software maintainability, (3) studies published before 2000, (4) short studies (2–3 pages), (5) secondary studies, and (6) studies by the same author; if results were the same in both studies, the most recent was used, otherwise both studies were used.

The study was retained if it satisfied both inclusion criteria, and rejected if it did not satisfy at least one of the exclusion criteria. Once applied, the decision to retain or reject the study depended on the evaluation of the two authors. In case of doubt or disagreement, a discussion based on review of the full text ensued until an agreement was reached. Duplicate titles and titles out of scope of the review were rejected.

2.4. Study quality assessment

Quality assessment (QA) criteria were used to assess the relevance of the candidate studies. QA is necessary in order to limit bias in conducting mapping and review studies, to gain insight into potential comparisons and to guide the interpretation of findings [12]. The quality of the relevant studies was evaluated based on seven questions as follows:

- **QA1**: Are the objectives of the study clearly described and appropriate?
- **QA2**: Are the factors or measures used as predictors of maintainability defined?
- **QA3**: Are the datasets adequately described?
- **QA4**: Are the SPMP techniques well-presented and defined?
- **QA5**: Is the accuracy criteria well-presented and discussed?
- **QA6**: Is the most accurate technique clearly stated?
- **QA7**: Are the findings of the study clearly stated and presented?

These questions have three possible answers: "Yes", "partially", and "No". These answers are scored as follows: (+1), (+0.5), and (0) respectively. The quality score for each study was computed by summing up the scores of the answers to the QA questions. The maximum score for all questions is 7 and the minimum 0. Studies that scored greater than 50% of the perfect score were considered for the review as in [4, 12]. The QA was performed independently by two of the authors. In the case of disagreement, the two authors discussed the issue until a final consensus was reached. After applying the QA criteria, 82 primary studies with an acceptable quality score (i.e. more than 3.5) were selected. The detailed quality scores for each study are presented in Table A3 in the Appendix.

2.5. Data extraction and data synthesis

A data extraction form was completed with information for each selected primary study to determine which apply to one of more of the mapping or research questions. Two independent researchers performed the extraction. In the case of disagreement, a discussion was held to reach consensus after a thorough review of the study. To facilitate synthesis and analysis of the data, the information collected was tabulated and grouped into a file (see Table 5). Various visualization techniques (such as charts and frequency tables, etc.) were used to synthesize the data, accumulate and combine facts from the selected primary studies in order to formulate answers to the mapping and research questions. A narrative summary reports the principal findings of the study, including collec-

Table 5. Data extraction form

Data extractor
Data checker
Study identifier
Name of database
Publication year
Author name(s)
URL
Article title
MQ2: Publication source
MQ3: Research type (see Table 3 and Table A1 in the Appendix)
MQ4: Empirical approach type (see Table 3 and Table A2 in the Appendix)
MQ5: Software application type (see Table 3)
MQ6: Datasets (see Table 3) – Categories of datasets – Historical datasets: name and number of projects
MQ7: Dependent and independent variables (see Table 3) – Common types of factors or measures used as independent variables (predictors). – Common types of factors or measures used as dependent variables. – Successful predictors of maintainability as reported in the selected primary studies. – Tools (tool name, description).
MQ8: Techniques (see Table 3) – Categories of techniques: statistical and machine learning.
RQ1: Prediction accuracy – Most used accuracy criteria. – Accuracy prediction of SPMP techniques per most used dependent variable topics (identified in MQ7).
RQ2: SPMP techniques reported to be superior in comparative studies – Techniques reported to be superior in comparative studies. – Strengths and weakness of these techniques. – Techniques having been reported to be superior and not.
RQ3: Accuracy comparison of the SPMP techniques identified in RQ2 – Selection of studies under the same prediction context (e.g. dataset, accuracy criteria, etc.). – Accuracy comparison of SPMP techniques under this context. – Selection of the most accurate SPMP techniques.

tion of a number of studies that state similar and comparable viewpoints.

3. Mapping Results

To conduct the study the process defined during the planning phase was implemented. Data retrieval, study selection, data extraction, and data synthesis were executed according to the review protocol developed by the authors. To begin with, the protocol was carried out by the first author in order to search studies related to the SPMP area. The first and second author then discussed the candidate studies after removing duplicates. Finally, the selected studies were checked by reading the full text of each study in order to confirm whether the paper was to be included or excluded from the list of primary studies. In cases of disagreement, the authors discussed the studies until an agreement was reached.

Figure 1 presents the search steps together with their corresponding results: (1) Applying the search terms on the nine online databases resulted in 41341 studies, (2) Removing duplicate studies and those not related to the SPMP topic resulted in 341 candidate studies, (3) Applying the inclusion and exclusion criteria resulted in 75 relevant studies, (4) Scanning the list of references and citations resulted in seven more studies for a total of 82 relevant studies (see Table A4 in the Appendix for the summary of the search results). All 82 studies were retained since they had an acceptable quality score (see Table A5 in the Appendix).

This section presents and discusses the results obtained from review of the 82 primary studies by providing answers to the mapping questions (MQ1-8) in the following subsections. The classification of each of the selected studies was based on the established classification criteria (see Table 3, and Tables A1 and A2 in the Appendix) and can be found in Table A6 in the Appendix.

3.1. Publication years (MQ1)

Figure 2 presents the distribution of SPMP studies per year, beginning in 2000. Interest in SPMP increased slowly over the decade from 2003 to 2010, reached a peak in 2012 and in 2017 (10 and 11 studies, respectively) and decreased thereafter while remaining relatively high between 2012 and 2017. Only three studies are shown for 2018 since most of the published studies were not yet online at the time the SMS was conducted.

3.2. Publication sources (MQ2)

Table 6 presents the distribution of the selected primary studies over publication sources. Only six journals (IJCA, JC, IST, IJSAEM, ESE, and JSS), six conferences (SIGSOFT, QR2MSE, ICSM, ICRITO, ICACCI, and CSMR) and one symposium (HASE) had more than one selected study. The other publication sources had only one study and have been grouped into others.

Figure 3 shows graphically the distribution of primary studies by source. Of the 82 selected

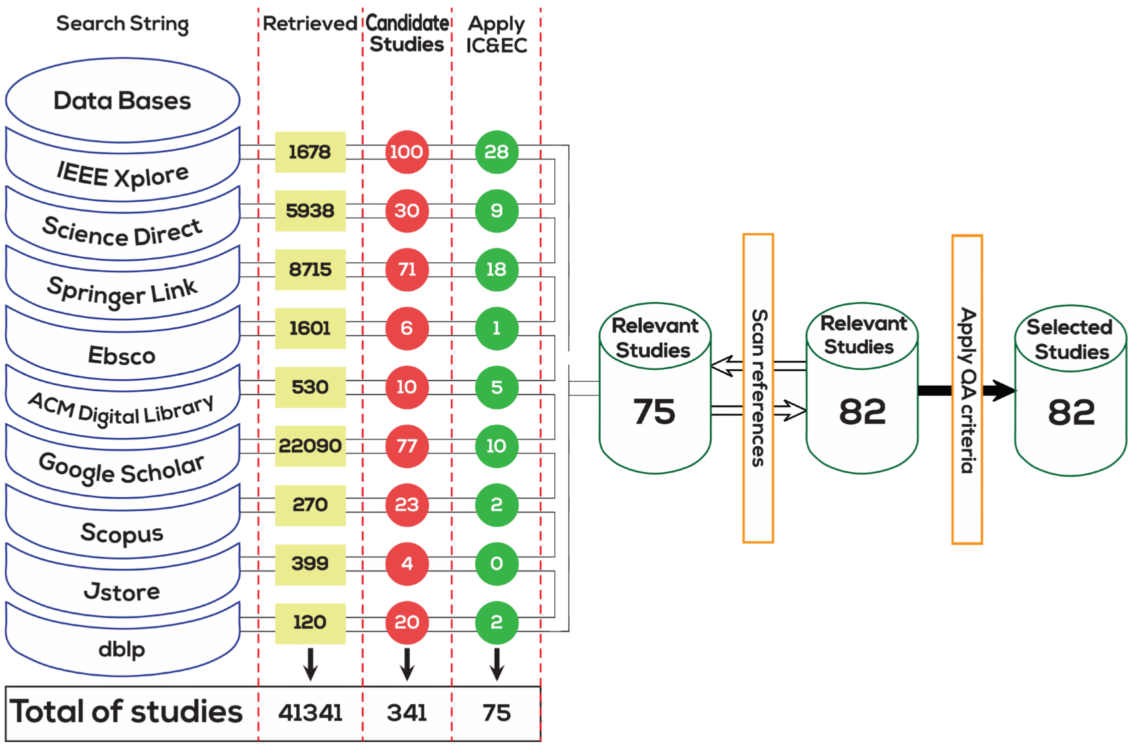


Figure 1. Search process steps and results

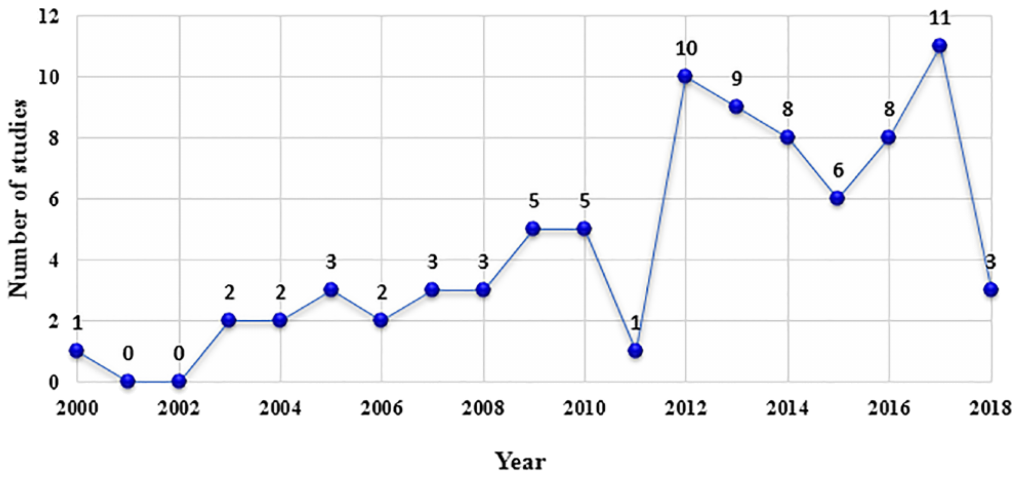


Figure 2. Distribution of selected SPMP studies per year

Table 6. Publication sources

Source	Type	#of studies	Percentage
Information and Software Technology (IST)	Journal	4	5%
Journal of Systems & Software (JSS)	Journal	4	5%
International Journal Computer Applications (IJCA)	Journal	3	4%
Empirical Software Engineering (ESE)	Journal	3	4%
Journal of Computing (JC)	Journal	2	2%
International Journal of System Assurance Engineering and Management (IJSAEM)	Journal	2	2%
SIGSOFT Software Engineering Notes (SIGSOFT)	Conference	2	2%
International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE)	Conference	2	2%
IEEE International Conference on Software Maintenance (ICSM)	Conference	2	2%
European Conference on Software Maintenance and Reengineering (CSMR)	Conference	2	2%
International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)	Conference	2	2%
International Conference on Advances in Computing, Communications and Informatics (ICACCI)	Conference	2	2%
International Symposium on High Assurance Systems Engineering (HASE)	Symposium	2	2%
Others (conference, symposium, journal, chapter, workshop)		1 each source	63%

studies, 41 (50%) were published in journals, 34 (42%) at conferences, four (5%) at a symposium, two (2%) in a workshop, and one (1%) a chapter.

3.3. Research types (MQ3)

Two main research types were identified from the selected studies: solution proposal (SP) and evaluation research (ER). Figure 4 shows that SP was the most frequently used (48 studies or 59%) followed by ER (34 studies or 41%), indicating that the goal of researchers was to propose new techniques or adapt old ones (SP), and then evaluate and/or compare existing techniques (ER) to improve SPMP. Furthermore, of the 82 selected studies, 41 (50%) conducted comparative studies to identify the most relevant techniques for

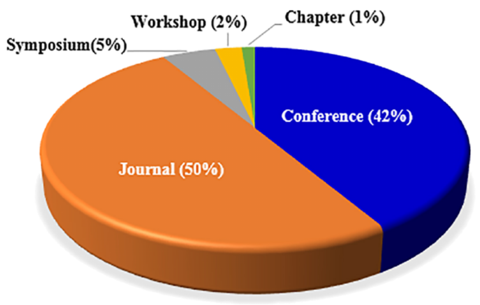


Figure 3. Distribution of primary SPMP studies by publication source

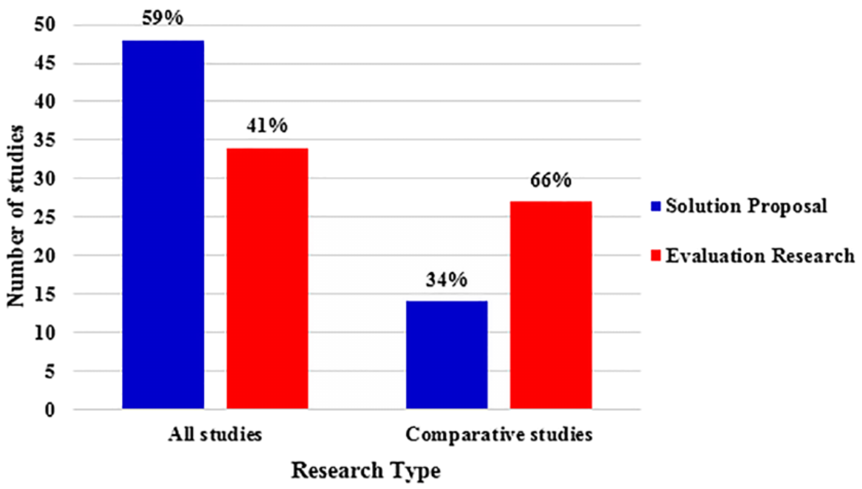


Figure 4. Research types of SPMP studies

predicting software product maintainability of which 14 (34%) were SP studies and 27 (66%) were ER studies.

3.4. Empirical approaches for validating SPMP techniques (MQ4)

Figure 5 shows the three main empirical approaches used to validate SPMP techniques, which are history-based evaluation (HbE), experiment (Ex), and case study (CS). From Figure 5, HbE and Ex were the most frequently employed approaches: 48 studies (58%) were empirically validated on previously completed software projects (HbE) and 26 studies (32%) were validated under controlled conditions (Ex).

As shown in Table 7, the number of studies using these two approaches has increased over time. Note that only eight out of 82 (10%) of selected studies investigated an SPMP technique in a real-life context through a case study (CS).

3.5. Software application types (MQ5)

To validate SPMP techniques, the selected studies used data from different types of software applications. A set of four main types were identified: object-oriented applications (OOA), procedure-oriented applications (POA), web-based applications (WbA), service-oriented applications (SOA), and component-based applications (CbA).

Figure 6 shows that OOA were the most frequently used with 65 studies (79%), followed by POA and SOA with four studies, each (5%), WbA with two studies (2%), and CbA with one study (1%). The remaining studies, denoted by NI (Not Identified), did not specify the type of software applications studied. The high percentage for OOA to empirically validate SPMP techniques is due to the use of historical datasets (MQ6), most of which involved object-oriented projects. Moreover, based on the distribution of primary studies using

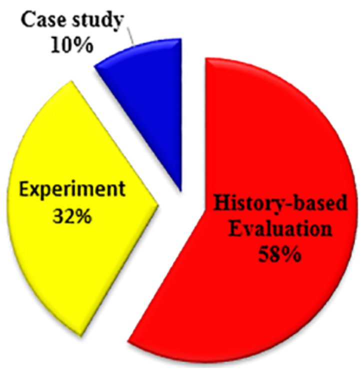


Figure 5. Empirical approaches for validating SPMP techniques

Table 7. Distribution of SPMP empirical approaches per time period

Empirical approach	2000–2005	2006–2011	2012–2018	Total
History-based evaluation (HbE)	2	10	36	48
Experiment (Ex)	5	4	17	26
Case Study (CS)	1	5	2	8

empirical approaches (MQ4) by software application type (MQ5), it can be seen in Figure 6 that OOA were frequently used in three empirical approaches: history-based evaluation (HbE) was the most frequently used, followed by experiment (Ex), and then case study (CS). Three other software application types were used less frequently: POA was only used in HbE and Ex approaches, WbA was used equally in CS and Ex approaches, SOA was only used in HbE, while CbA was only used in CS.

Figure 7 shows the frequency of research types (MQ3), empirical approaches (MQ4) and software applications types (MQ5). It can be remarked that:

- OOA were the most frequently studied in both research types (31 for evaluation research and 34 for solution proposal),
- POA, WbA, SOA, and CbA software application types were less used (eight studies for solution proposal research), and
- the remaining six studies did not clearly identify the software applications types considered.

Moreover, almost all evaluation research studies (31 of 34 studies) used the HbE evaluation approach while the majority of solution proposal studies used either Experiment (23 studies) or HbE evaluation (17 studies) approaches. The case study approach was less used and only in solution proposal (eight studies).

3.6. Datasets (MQ6)

A variety of datasets from various sources were used in the selected the primary studies. Three main categories of datasets based on their origin were identified:

- Software engineering researchers (SER): Public datasets used by researchers from the software engineering community: UIMS (user interface management system), QUES (quality evaluation system), VSSPLUGIN (visual source safe PLUGIN), PeerSim (peer-to-peer simulator), etc.
- Open source software systems/projects (OSS): Freely available datasets, such as JHotdraw, Jtreeview (Java TreeView), JEdit, Lucene, etc.
- Private software projects/systems (PSP): Private data from large industrial projects, such as: MIS (medical imaging system), FLM (file letter monitoring system), EASY (EASY classes online services collection), SMS (student management System), IMS (inventory management system), APB (angel bill printing), and from academic software projects

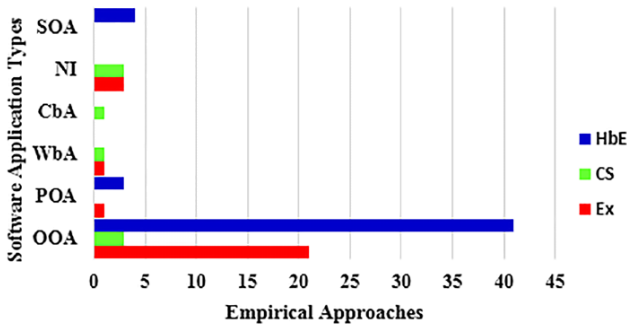


Figure 6. Frequency of software application type per empirical approach

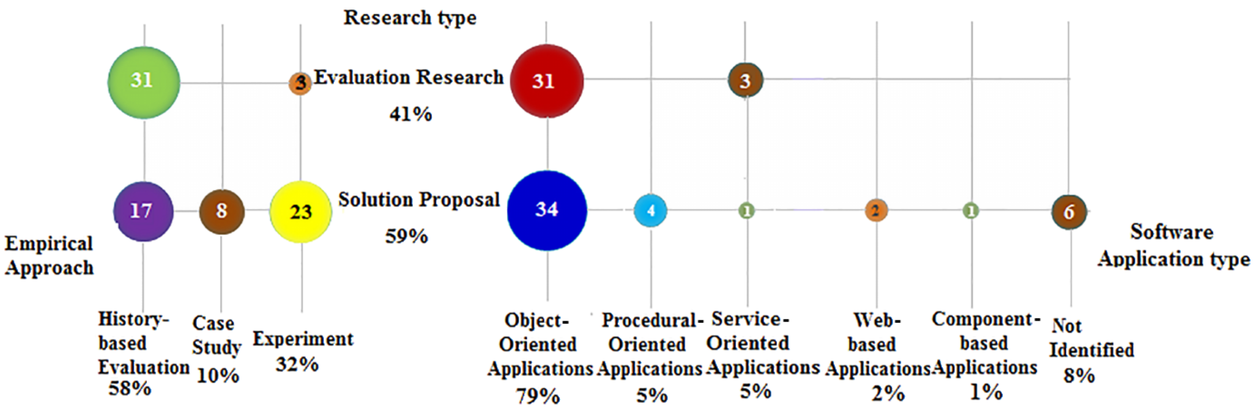


Figure 7. Frequency of research types, empirical approaches and software application types

Table 8. Number of SPMP studies per dataset sources

Dataset sources	Used in	# of studies	Percent
Private software projects (PSP)	S1, S3, S4, S7, S10, S11, S12, S13, S15, S17, S18, S20, S22, S24, S25, S30, S34, S36, S40, S47, S50, S51, S52, S53, S67, S74, S75, S78, S79, S80, S81, S82	32	39%
Open sources software projects (OSS)	S5, S8, S16, S27, S28, S34, S36, S39, S41, S46, S48, S49, S59, S60, S61, S62, S63, S64, S65, S66, S68, S70, S71, S72, S73, S76, S77	27	33%
Software engineering researchers (SER)	S2, S6, S9, S14, S19, S21, S23, S26, S29, S31, S32, S33, S35, S37, S38, S42, S43, S44, S45, S54, S55, S57, S56, S58, S69	25	30%

developed by students, such as bank information system (BIS) and Aggarwal datasets.

Table 8 presents the number and percentage of studies per dataset sources. The PSP datasets were the most frequently used with 32 studies (39%) each, followed by OSS datasets with 27 studies (33%) and SER datasets with 25 studies (30%). Note that some studies may have used more than one dataset. For example, S34 used both PSP and OSS datasets and was counted twice.

Within these dataset sources, some empirical studies used historical data to evaluate and/or compare SPMP techniques with other techniques, referred to as historical datasets. When researchers collect data on their own, they can make it available for future use or not. When the available data is used by other research workers, it is referred to as historical datasets. From the set of 82 selected studies, 48 (which are related to HbE (MQ4)) used historical datasets. Table 9 summarizes the historical datasets used, the number and percentage of

Table 9. Distribution of HbE empirical approaches over historical datasets

Datasets	# of studies	Percent	# of project	Source
UIMS	24	29%	1 project (39 classes)	[13]
QUES	22	27%	1 project (71 classes)	[13]
JEdit	6	7%	1 project (415 classes)	[14]
eBay	4	5%	1 projet (1524 classes)	[15]
Lucene	3	4%	1 project (385 classes)	[14]
JHotdraw	3	4%	1 project (159 classes)	[14]
Art of Illusion	3	4%	1 project (739 classes)	[16]
jTDS	3	4%	1 project (64 classes)	[17]
BIS	2	2%	1 project (28 classes)	[18]
MIS	2	2%	1 project (4500 modules)	[19]
JUnit	2	2%	1 project (251 classes)	[20]
Ivy	2	2%	1 project (614 classes)	[16]
Camel	2	2%	1 project (422 classes)	[16]
Eclipse	2	2%	1 project (10 594 classes)	[16]
FLM	2	2%	1 project (55 classes)	[21]
EASY	2	2%	1 project (84 classes)	[21]

the primary studies that used the dataset, the number of projects or classes and the source reference of the dataset. Note that one study may involve more than one dataset and in that case is counted only once. As can be seen from Table 9, among the 48 HbE empirical approaches, the most frequently used historical dataset was UIMS (24 studies) followed by QUES (22 studies), which amounts to 56% for only two relatively small OOA datasets of one project each. While this creates a limitation in terms of bias in the evaluation of numerous studies, it permits a basis for comparison across findings using the same dataset. Datasets that were used in two to four studies included: JEdit, eBay, JHotdraw, jtds, Lucene, Art of Illusion, Eclipse, MIS, FLM, BIS, Ivy, Junit, Camel, and EASY. The remaining datasets were used in only one study each (not included in Table 9).

Furthermore, all these datasets (4th column) developed software projects using the object-oriented paradigm (including classes, methods, attributes, polymorphism, etc.), except MIS and Aggarwal datasets which developed software projects using the procedure-oriented paradigm (POA) and eBay software applications using the service-oriented paradigm.

Figure 8 is extracted from Table 9 and includes only software engineering researchers and open source datasets from publicly available industrial or professional datasets, such as: UIMS, QUES, JEdit, Lucene, JHotdraw (no private or student datasets were included). For instance, the two popular datasets published by Li and Henry [13] (UIMS and QUES), which are frequently used in predicting maintainability, are OO commercial systems developed using the Ada programming language. The other datasets (JEdit, Lucene, JHotdraw, Art of Illusion, jTDS, JUnit, Ivy, Camel, Eclipse, and eBay) are OO systems implemented in Java. The public availability of these datasets allows researchers and practitioners to conduct verifiable, repeatable, comparatives studies [22], provided that they use the same prediction context (e.g. dependent and independent variables, datasets, accuracy criteria, and validation method).

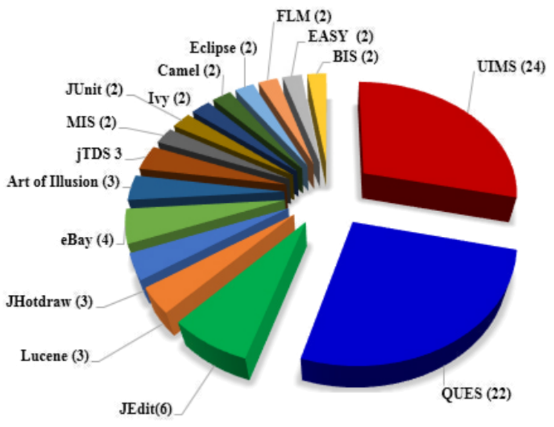


Figure 8. Historical datasets used for SPMP studies

Table 10. Classification of the dependent variables

Topic	Sub-topic	Supported studies	# of studies
Change	Changes in LOC	S2, S6, S9, S14, S19, S21, S23, S24, S26, S29, S31, S32, S33, S35, S37, S38, S42, S43, S44, S45, S47, S48, S49, S52, S54, S55, S56, S57, S58, S59, S60, S61, S62, S64, S65, S66, S69, S70, S71, S72, S73, S74, S77	46
	Change in module	S10, S15	
	Change in class	S30	
Expert opinion	Expert opinion based on ordinal scale	S11, S18, S20, S25, S27, S28, S36, S41, S50, S67	10
Maintainability index	Maintainability index	S8, S16, S48, S68, S75	8
	Relative maintainability index	S39, S76	
	Maintainability index satisfaction	S40	
Maintainability level	Understandability level, modifiability level, analyzability level	S4, S22, S51, S53, S78, S80, S82	7
Maintainability time	Understandability time	S3, S12, S78	8
	Modifiability time	S3, S12, S78	
	Completion time of understandability	S80, S82	
	Time to repair the design of a structure	S17	
Other	Maintainability expressed in terms of number of revised lines of code and number of revisions	S46, S63	2
	Maintainability efficiency	S79	1
	Maintainability effectiveness	S79	1
	Understandability effectiveness	S81	1
	Modifiability effectiveness	S81	1
	Understandability efficiency	S81	1
	Modifiability efficiency	S81	1
	Modifiability completeness	S3	1
	Modifiability correctness	S3	1
	Error prone modules	S1	1
	Detected fault	S13	1
	Maintainability measured using probabilistic quality model	S34	1
	WbA maintainability	S5	1
	Perceived maintainability	S7	1

3.7. Dependent and independent variables used in SPMP studies (MQ7)

This section identifies and discusses the dependent variables and the measures used to express maintainability (predicted output). It then presents the factors or measures used as independent variables (predictors), the tools used to gather them and the reported successful predictors of software product maintainability from the 82 primary studies.

3.7.1. Dependent variables

The dependent variable (predicted output), software maintainability, was measured differently in the 82 selected studies. As shown in Ta-

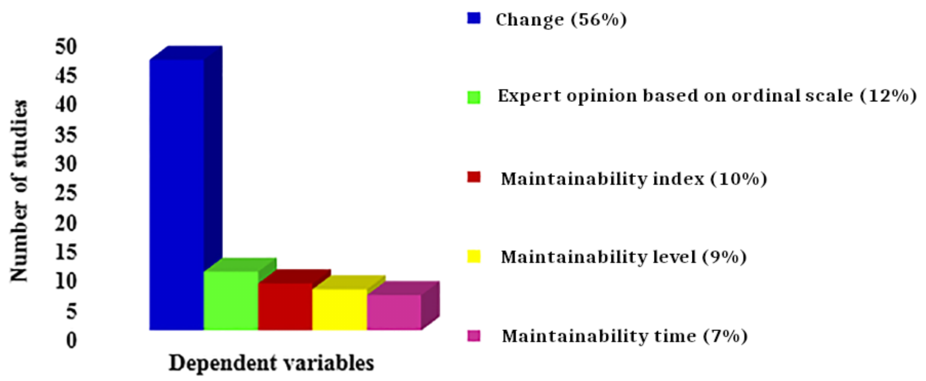


Figure 9. Distribution of selected SPMP studies per most used dependent variable

ble 10 and Figure 9, we identified five main research topics related to maintainability (or its sub-characteristics). Other less used research topics were also identified, but are not included in Figure 9. The scope of this review included the maintainability sub-characteristics as identified by ISO 9126 [23] or its successor ISO 25010 [24], such as: changeability, modifiability, stability, analysability, testability, modularity, and reusability, or as defined by a particular study (S4, for example, identified two sub-characteristics of maintainability: understandability and modifiability).

As shown in Table 10:

- The topic most frequently referred as the dependent variable is change, 46 studies (56%):
  - Changes in LOC studies used the number of lines of code changed per class by counting the number of lines in the code that were changed.
  - Changes of modules studies used the changes made to each module due to faults discovered during system testing and maintenance.
  - Changes of classes studies used the change of an attribute, a method or a class affected by decomposition of the system and its sub-systems.
- The second topic referred to studies that predict SPM based on expert opinion: 10 studies (12%) expressed maintainability using an ordinal scale based on expert opinion. The maintainability was qualified as: poor, average, very good, or very high, high, medium, low, or excellent, average, bad, etc.
- The third topic referred to studies that used a maintainability index (MI) to determine the maintainability of the software product (eight studies – at 10%). Some studies used the maintainability index calculated as a factored formula of average Halstead volume per module, average extended cyclomatic complexity, average lines of code, and average percent of lines of comments per module measures. Some studies used relative maintainability index calculated for each source code element for which metrics were calculated (e.g. methods, classes) using the goodness value. Other studies used the maintainability index satisfaction expressed in terms of maintenance time satisfaction, maintenance man-hour satisfaction, and maintenance cost satisfaction.
- The fourth topic referred to studies that predict maintainability in terms of understandability, modifiability and analyzability levels, which are evaluated based on the subject's difficulty to: understand the system, carry out modification tasks, and diagnose the system (seven studies – 9%).
- The fifth topic refers to studies that predict the maintainability in terms of understandability time, and/or modifiability time spent by subjects answering the understandability questions or understanding source code and carrying out modifications, or the time to repair the design of structure (six studies – 7%).
- The other research maintainability topic included less used dependent variables such as: modifiability completeness, modifiability correctness, number of revised lines of code

and number of revisions, maintainability efficiency, maintainability effectiveness, understandability effectiveness, modifiability effectiveness, understandability efficiency, modifiability efficiency, error prone modules, detected fault, maintainability measured using a probabilistic quality model, WbA maintainability, and perceived maintainability.

3.7.2. Independent variables

In order to predict software product maintainability, the selected primary studies used various factors or measures as independent variables (or predictors), i.e. different inputs to the SPMP techniques. This subsection presents the independent variables used, those most used as predictors and the tools used to collect them. For the remainder of this paper, the terms independent variables and predictors will be used interchangeably. Table A7 in the Appendix provides the full list of the predictors used, the corresponding total number of frequencies, supported studies and percentage.

For the 82 primary studies, Chidamber and Kemerer (C&K) measures were the most used (50 studies – 61%), followed by

- Li and Henry (L&H) measures (33 studies – 40%),
- Class diagram measures (24 studies – 29%), which included measures related to method, attribute, class, or relationships (associations, aggregations, generalization and dependency),
- Source code size measures using different lines of code (LOC) measures (20 studies – 24%),
- McCabe cyclomatic complexity (17 studies – 21%), and
- Software quality attributes (such as stability, changeability and analyzability, readability of source code, document quality, understandability of software, simplicity, accessibility, etc.) (eight studies – 10%).

The least used predictors included measures such as: factors, Lorenz and Kidd (L&K) measures, coding rule measures, maintainability index (MI), web-based application (WbA), sequence diagram measures (scenarios, messages and conditions), Martin's measures, QMOOD measures, Fault, database measures, Halstead measures, and Brito e Abreu and Carapuça (BA&C), etc.

Figure 10 shows the number of studies for the most frequently used predictors. Note that one study may involve more than one type of predictor. Figure 10 is extracted from Table A7, while the least used predictors were discarded.

Furthermore, it was observed that object-oriented measures were the most used predictors. This is mainly due to the wide use of object-oriented software applications (OOAs) in SPMP empirical studies, i.e. 65 out of the 82 selected studies (see Section 3.5, Figures 6 and 7).

As shown in Figure 11, the frequently used OO measures were RFC (response for a class) and LCOM (lack of cohesion in methods), followed by WMC (weighted methods per class), DIT (depth of inheritance tree), NOC (number of children), LOC (lines of code or size1), MPC (message passing coupling), NOM (number of local methods), DAC (data abstraction

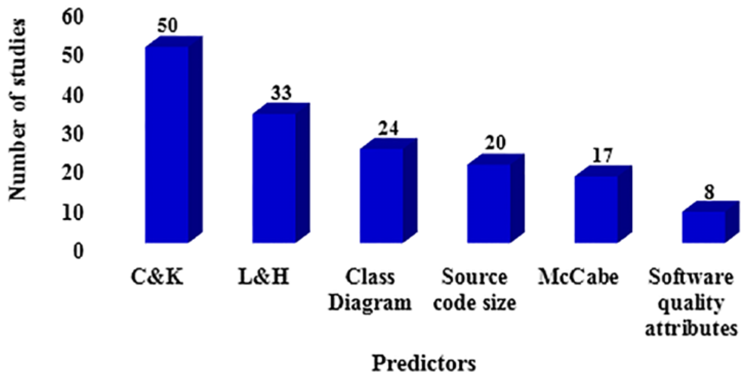


Figure 10. The number of the SPMP studies for the most frequently used predictors

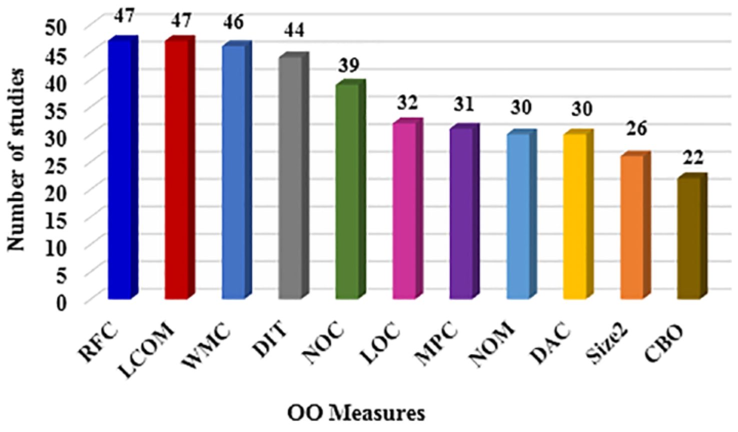


Figure 11. The number of SPMP studies for the most frequently used OO measures

coupling), Size2 (number of properties), and CBO (coupling between object). Such types of measures were collected at the design or source code levels.

Table 11 presents the list and description of the tools used to compute these measures, as well as the primary studies that used them. Note that only 46 out of the 82 studies provided information on the data collection tools used. The Classic-Ada metrics analyzer was the most commonly used (24 studies), followed by Chidamber and Kemerer Java Metric (CKJM) tool (eight studies), Intellij IDEA tool (four studies), LocMetrics tool (three studies), Krakatau Professional tool and Understand tool (two studies each), and one study each for Columbus tool, web application reverse engineering (WARE) tool, Analyst4j standalone tool, COIN tool, ObjectAid UML Explorer, JHawk tool, JDepend tool, Classycle tool, SourceMeter static code analysis tool, Customized tool, and C and C++ code counter (CCCC) tool. Four other studies used their own private tools.

Table 11. Tools used to collect measures

Name	Description	ID
Classic-Ada metrics analyzer	Classic-Ada was implemented in LEX and YACC UNIX environments and was designed on the Mach operating system running on a NeXTstation using a GNU C compiler. The system was ported to an Ultrix system running on a VAX station [13].	S2, S6, S9, S14, S19, S21, S23, S26, S29, S31, S32, S33, S35, S37, S38, S42, S43, S44, S45, S54, S57, S56, S58, S69
CKJM	Chidamber and Kemerer Java Metric extraction tool is freely available. It calculates C&K metrics by processing the bytecode of Java files [25].	S48, S49, S55, S59, S61, S66, S71, S72
Intellij IDEA	Intellij IDEA is a free and open source Java IDE developed by JetBrains and available as Apache 2 licensed and community edition [26].	S48, S49, S62, S64
LocMetrics	LocMetrics[1] counts total lines of code, blank lines of code, comment lines of code, lines with both code and comments, logical source lines of code, McCabe VG complexity, and number of comment words	S66, S71, S72
Krakatau Professional	Krakatau Professional was developed by Power Software Inc. It is a fully-featured software metrics tool designed for source code quality and software measurement specialists [27].	S8, S41

Table 11 continued

Name	Description	ID
Understand	Understand[2] is very efficient at collecting metrics about the code and providing different ways for you to view it. There is a substantial collection of standard metrics available as well as options for writing custom metrics.	S73, S77
Columbus	Columbus is a framework that supports project handling, data extraction, data representation, data storage and filtering [28].	S39
WARE	WARE is an integrated tool that automatically extracts information from the application and allows more abstract representations to be reconstructed [29].	S5
COIN	Cohesion Inheritance (COIN) is a tool for evaluating cohesion, inheritance and size metrics of class hierarchies in Java projects [30].	S68
Analyst4j standalone tool	Analyst4j is based on the Eclipse platform. It features search, metrics, analyzing quality, and report generation for Java programs [31].	S28
ObjectAid UML Explorer	ObjectAid UML Explorer[3] has been used to extract the UML diagrams from the Java source code.	S63
JHawk	JHawk[4] is a general-purpose metrics collection tool that calculates a variety of metrics from OO systems.	S63
JDepend	JDepend[5] has been used to generate design quality metrics for each package in the system and verify the relations between the packages.	S63
Classycle	Classycle's Analyser tool[6] analyzes the static class and package dependencies in Java applications.	S63
SourceMeter	SourceMeter[7] is an innovative tool built for precise static source code analysis of C/C++, Java, C#, Python, and RPG projects. This tool makes it possible to find the weak spots of a system under development from the source code only, without the need to simulate live conditions.	S76
CCCC	CCCC[8] is a free software tool by Tim Littlefair for measurement of source code related metrics.	S52
Customized tools	Customized tools have been implemented to integrate and analyze data from previous tools and to compute the new coupling, instability and abstractness metrics for every package in the system [32].	S63
Private	Tools constructed and developed for each study according to the context to automatically collect metrics.	S4, S5, S46, S52

[1]http://www.locmetrics.com
[2]http://www.scitools.com
[3]http://www.scitools.com
[4]http://www.virtualmachinery.com/jhawkprod.htm
[5]http://clarkware.com/software/JDepend.html
[6]http://classycle.sourceforge.net/
[7]http://www.sourcemeter.com/
[8]http://cccc.sourceforge.net/

Regarding successful predictors of SPM, 25 (30%) of the 82 selected studies explicitly reported useful measures for software product maintainability based on empirical evaluation – see Table 12:

– Chidamber & Kemerer and Li & Henry measures (DIT, NOC, WMC, RFC, CBO, LCOM, MPC, DAC, NOM, SIZE1, and SIZE2) reported good correlation with maintainability in 14 studies (17%).

Table 12. Successful predictors of SPM in 25 of the SPMP studies

Successful predictors	Supported by
DIT, NOC, WMC, RFC, CBO, LCOM, MPC, DAC, NOM, SIZE1, SIZE2	S6, S8, S9, S14, S21, S32, S46, S47, S48, S49, S52, S58, S60, S66
NA, NM, NC, NAgg, NAggH, NGen, NGenH, NAssoc, NDep, MaxDIT	S3, S12, S22, S51, S68
MI, CC, NODBC, SCCR	S52, S68
TWP, TLOC, WO, SS, ClS, TL, TCC, TWPR, TWPDC	S5, S68
Coding effort, RDCRatio	S7
Average fan-out, data flow, average McCabe	S1
ACLOC, AMLOC, AVPATHS, CDENS, COF, n, N, PPPC	S8
NPAVGC, OSAVG, CSA, SWMC, POF	S16
LLOC, McCabe, rule violations	S39
NOA, Coh, CAMC, LCC, LSCC, SCOM, PCCC, OL2, CBO_U, CBO_IUB, OCMEC, TCC	S46, S68
B, CLOC, Command, CONS, CSA, CSO, Cyclic, Dcy, NAA, OCmax, OSmax, SLoc, STAT, V, Query	S48
B, CLOC, Command, Inner*, Dcy*	S49
NclienP, NAggR, NAssoc, NservP, NwebP	S53
LCOM3, LOC, Ce	S60
NPM, Ca, DAM, MOA	S66
MIF, AIF, DCi, Coh, DCd	S68

– Class diagram measures (NA, NM, NC, NAgg, NGenH, NAssoc, NDep, MaxDIT, NGen, and NAggH) were found to be useful in predicting maintainability in five studies (6%).
– The other measures were reported useful in two or one study each.

The remaining 36 studies did not report useful measures, since most were interested in comparing the accuracy of their proposed or evaluated SPMP techniques rather than in identifying successful predictors. See Table A8 in the Appendix for the acronyms of the successful predictors.

3.7.3. Summary

Table 13 presents the predictors (independent variable) used by each maintainability research topic.

– Studies focusing on predicting maintainability in terms of change used mainly C&K and L&H measures, and in particular, change expressed in terms of number of LOC changed in a class. This was because the datasets used (e.g. UIMS, QUES, FLM, EASY, and Lucene, etc.) focused on OO software applications.
– Studies on maintainability based on expert opinion using an ordinal scale used quality attributes, such as readability of source code, document quality, stability, changeability, analyzability as dependent variable, or measures related to source code size, McCabe, C&K, class and coding rules.
– Studies on maintainability index or relative maintainability index used C&K, source code size, Halstead, class, Lorenz and Kidd, Brito e Abreu and Carapuça, and McCabe measures, while the maintainability index satisfaction used satisfaction attributes.
– Studies on maintainability level in terms of sub-characteristics (understandability, modifiability and analysability) used class diagram as well as sequence diagram measures and factors.
– Studies on maintainability time used class diagram measures for understandability time and modifiability time, while some used software quality attributes.
– Most of the remaining topics used class diagram, source code size, as well as factors and McCabe measures.

Furthermore, some studies, including S6 and S8, reported that C&K and L&H measures (which

Table 13. Type of independent variable by dependent variable topic

Topic (dependent variable)	Predictor measures (independent variables)
Change	C&K, L&H, McCabe, maintainability index, database, class, Halstead, source code size, Jensen, effort
Expert opinion based on ordinal scale	Quality attributes, source code size, McCabe, coupling, C&K, class, coding rules
Maintainability index	C&K, source code size, Halstead, class, Lorenz and Kidd, Brito e Abreu and Carapuça, McCabe, quality attributes
Maintainability level	Class diagram, sequence diagram class
Maintainability time	Class diagram, quality attributes
Modifiability correctness	Class diagram, factors
Modifiability completeness	Class diagram, factors
Maintainability efficiency	factors
Maintainability effectiveness	factors
Understandability effectiveness	factors
Modifiability effectiveness	factors
Understandability efficiency	factors
Modifiability efficiency	factors
Error prone modules	McCabe, module level
Detected fault	fault

are related to OO design attributes such as coupling, cohesion and inheritance) were statistically significant and highly correlated to maintainability. Note also, that C&K and L&H measures as predictors were most often used to predict maintainability expressed in terms of change as predicted output.

3.8. Techniques used in SPMP studies (MQ8)

From the 82 selected primary studies we identified two major categories of techniques that have been applied to predict software product maintainability: machine learning (ML) and statistical techniques. Figure 12 shows that ML techniques were the most frequently used, being adopted by 70% (57 studies) compared to statistical techniques with 51% (42 studies). Note that we include all studies using single techniques in the review results section. Note, too, that a study may use techniques from the two categories (more details in Table A6 in the Appendix).

The statistical techniques include regression analysis (RA), probability density function (PD), gaussian mixture model (GMM), discriminant analysis (DA), weighted functions (WF) and stochastic model (SM):

- **RA** were the most frequently used statistical techniques with 35%. This category includes: Linear Regression (LR), Multiple Linear Regression (MLR), Logistic Regression (LgR), Backward Elimination (BE), Stepwise Selection (SS), Multiple Adaptive Regression Splines (MARS), Projection Pursuit Regression (PPR), polynomial regression (PR), Least Median of Squares Regression (LMSR), Pace Regression (PaceR), Isotonic Regression (IR), Regression by Discretization (RegByDisc), Additive Regression (AR), Gaussian Process Regression (GPR), and Least Absolute Shrinkage and Selection Operator (Lasso), followed by
- **PD** with 4%, **SM**, **GMM**, **DA** and **WF** with 1% each.

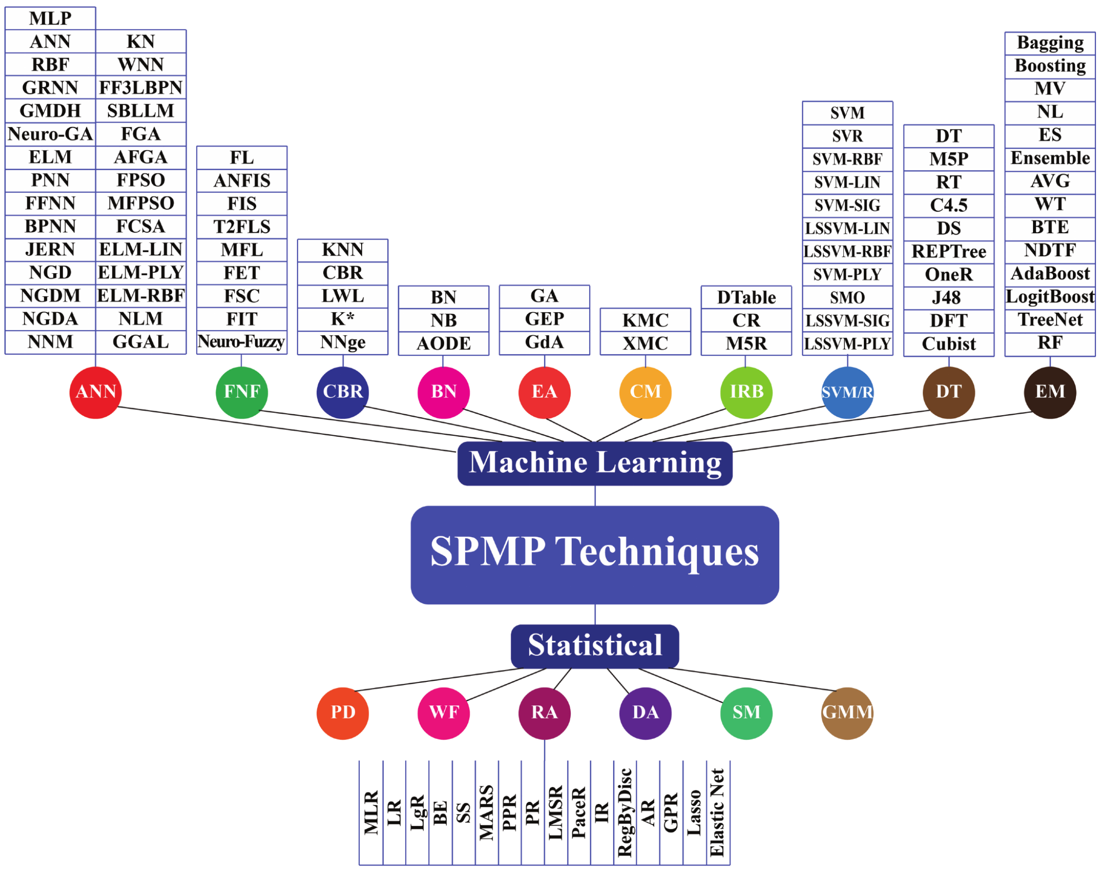


Figure 12. Techniques used in SPMP studies

The ML techniques were categorized according to [33, 34] as follows: Artificial Neural Network (ANN), Fuzzy & Neuro Fuzzy (FNF), Regression & Decision Trees (DT), Ensemble Methods (EM), Case-Based Reasoning (CBR), Bayesian Networks (BN), Evolutionary Algorithm (EA), Support Vector Machine & Regression (SVM/R), Inductive Rule Based (IRB), and Clustering Methods (CM).

- **ANN** were the most used techniques with 38%. It includes Multilayer Perceptron (MLP), Radial Basis Function Network (RBF), Probabilistic Neural Network (PNN), Group Method of Data Handling (GMDH), General Regression Neural Network (GRNN), Feed Forward Neural Network (FFNN), Back Propagation Neural Network (BPNN), Kohonen Network (KN), Ward Neural Network (WNN), Feed Forward 3-Layer Back Propagation Network (FF3LBPN), Extreme Learning Machines (ELM), Sensitivity Based Linear Learning Method (SBLLM), Neuro-Genetic Algorithm (Neuro-GA), Functional Link Artificial Neural Network (FLAAN) with Genetic Algorithm (FGA), Adaptive FLANN-Genetic (AFGA), FLANN-Particle Swarm Optimization (FPSO), Modified-FLANN Particle Swarm Optimization (MFPSO), FLANN-Clonal Selection Algorithm (FCSA), ELM with Linear (ELM-LIN), ELM with Polynomial (ELM-PLY), ELM with Radial Basis Function Kernels (ELM-RBF), ANN with Levenberg Marquardt Method (NLM), GRNN with Genetic Adaptive Learning (GGAL), Jordan Elman Recurrent Network (JERN), ANN with Normally Gradient Descent Method (NGD), ANN with Gradient Descent With Momentum (NGDM), ANN with Gradient Descent With Adaptive Learning Rate (NGDA) and ANN with Quasi-Newton Method (NNM).
- **SVM/R** with 24%, includes Support Vector Machine (SVM), Support Vector Regression (SVR), Sequential Minimal Optimization (SMO), SVM with Radial Basis Func-

tion Kernel (SVM-RBF), SVM with Linear Kernel (SVM-LIN), SVM with Sigmoid Kernel (SVM-SIG), Least Square Support Vector Machine (LSSVM) with Linear Kernel (LSSVM-LIN), LSSVM with Radial Basis Function Kernel (LSSVM-RBF), SVM with Polynomial Kernel (SVM-PLY), LSSVM with Sigmoid Kernel (LSSVM-SIG) and LSSVM with Polynomial Kernel (LSSVM-PLY).

- **FNF** with 20%, includes Fuzzy Logic (FL), Adaptive Neuro-Fuzzy Inference Systems (ANFIS), Fuzzy Inference Systems (FIS), Type-2 Fuzzy Logic System (T2FLS), Mamdani-based Fuzzy Logic (MFL), Fuzzy Entropy Theory (FET), Fuzzy Subtractive Clustering (FSC), Fuzzy Integral Theory (FIT), and Neuro-Fuzzy.
- **DT** with 18%, includes Regression Tree (RT), M5 For Inducing Trees of Regression Models (M5P), Decision Stump (DS), Reduced Error Pruned Tree (REPTree), Decision Tree Forest (DFT), C4.5, OneR, J48, and Cubist.
- **EM** with 15%, includes Ensemble Selection (ES), Average-based Ensemble (AVG), Weighted-based Ensemble (WT), Best-in-Training-based Ensemble (BTE), Majority-Voting Ensemble (MV), Non-Linear Ensemble (NL), Nonlinear Ensemble Decision Tree Forest (NDTF), Adaptive Boosting (Adaboost), Bagging, Boosting, Ensemble, Random Forest (RF), TreeNet, and LogitBoost.
- **BN** with 7%, includes Naive-Bayes (NB) and Aggregating One-Dependence Estimators (AODE).
- **CBR** with 6%, includes Kstar (K*), Locally Weighted Learning (LWL), *k*-Nearest Neighbor (IBK or KNN), and Nearest-Neighbor-Like algorithm that uses Non-Nested generalized exemplars (NNge).
- **EA** with 6%, includes Genetic Expression Programming (GEP), Genetic Algorithm (GA) and Greedy Algorithm (GdA).
- **IRB** with 4%, includes Decision Table (Dtable), Conjunctive Rule Learner (CR), and M5 Rules (M5R).
- **CM** with 2%, includes K-Means Clustering (KMC) and x-Means Clustering algorithm (XMC).

4. Review Results

This section presents and discusses the results of this review by providing answers to the three research questions (RQ1-3) in Table 2. Through these questions, the following subsections analyze the SPMP techniques from three perspectives: prediction accuracy, techniques reported superior in comparative studies and accuracy comparison of the techniques. Note that only studies with consistent results about accuracy have been taken into account, thereby excluding S56.

4.1. Prediction accuracy (RQ1)

From the results of MQ7, change, expert opinion, maintainability index, maintainability level, and maintainability time were the most used dependent variable topics (i.e. measures used to express maintainability, the predicted output) from a set of 74 selected SPMP studies. Table A9 in the Appendix shows the details of the SPMP techniques, the accuracy criteria used, and the mapping to the corresponding studies, grouped by the most addressed dependent variable topics. As can be seen, different accuracy criteria were used such as: mean magnitude of relative error (MMRE), percentage relative error deviation (Pred(25) and Pred(30)), coefficient of correlation R, Coefficient of determination (R-squared), root mean square error (RMSE), normalized RMSE (NRMSE), mean absolute error (MAE), mean absolute relative error (MARE), magnitude of relative error (MRE), accuracy, precision, weighted average precision (WAP), recall, F-measure, specificity, etc., where MMRE, Pred(25) and Pred(30) were the most dominant. MMRE measures the mean of the difference between the actual and the predicted value based on the actual value, while Pred measures the percentage of predicted values that have an MRE less than or equal to 0.25 or 0.30 [3].

Note that we included studies that used MMRE and/or Pred to evaluate prediction accuracy in this research question. Topics for which there was no MMRE or Pred were discarded. Note too, that low MMRE or high Pred(25) or Pred(30) values indicated good prediction accuracy [35, 36].

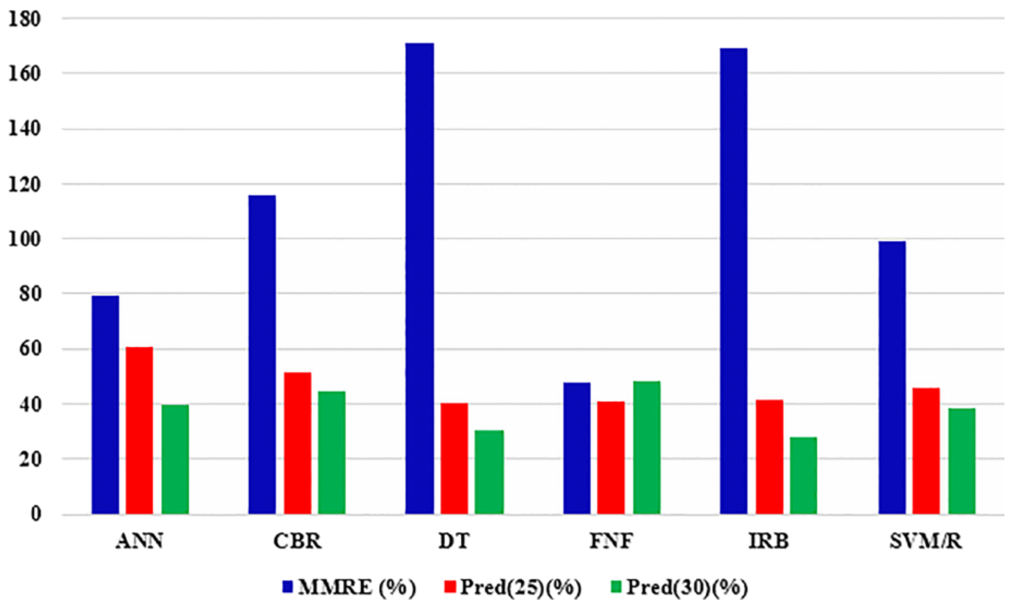


Figure 13. Average performance of different change prediction techniques (16 studies)

Change: Selected studies on the change topic (including changes of lines in the code, or changes made to each module, or changes of an attribute, a method or a class to predict the maintainability of a software) used MMRE, Pred(25), and Pred(30) in 16 out of 44 studies as accuracy criteria. We also looked into the average performance of the different prediction techniques. As shown in Figure 13, FNF had the lower value in terms of MMRE and the highest value in terms of Pred(30), ANN had the highest value in terms of Pred(25). Moreover, FNF provided greater accuracy in terms of MMRE and Pred(30). The remaining studies (24 out of 44) used different accuracy criteria such as R-squared, R, MAE, MARE, RMSE, NRMSE, precision, recall, F-measure, specificity, accuracy, etc., while four studies did not provide the accuracy criteria used (see Table A9 in the Appendix for more details).

Maintainability index: Eight studies used the maintainability index for prediction accuracy. Most studies under this topic used various accuracy criteria such as: coefficient of correlation, R-squared, adjusted R-squared, standard error of the estimate and Spearman's coefficient of correlation (Rs), etc. Only study S68 used MMRE, and Pred(30), while study S16 used MMRE as accuracy criteria. Note that a set of 105 experiments were performed in S68 and S16.

The distribution of prediction performance of these two studies is shown in Figure 14 in terms of MMRE and Pred(30). The MMRE ranged from 1% to 100%, while the Pred(30) varied from 40% to 100%.

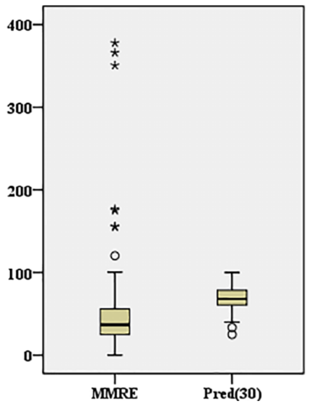


Figure 14. Performance distribution of maintainability index (S16 and S68)

Maintainability time: All studies (8) under this topic predicted maintainability in terms of understandability time, and/or modifiability time while performing tasks related to maintainability. Accuracy was evaluated using various accuracy criteria such as: R-squared and qMRE, etc. One study (S3) used MMRE and Pred(30) as accuracy criteria in three experiments and the RA (MLR) technique to predict maintainability time.

Table 14. Prediction performance for maintainability time

				Prediction context		
ID	MMRE	Pred(30)	Dataset type	Software developement project	Dependent variable	Prediction technique
S3	58.30	46.00	Spain data	Object-oriented	Understandability time	RA
S3	67.60	38.50	Italy data	Object-oriented	Understandability time	RA
S3	85.00	30.00	All data	Object-oriented	Understandability time	RA

Table 14 shows its prediction accuracy as well as prediction context. The average MMRE was 70% and the average Pred(30) was 38%. The result shows that the experiment using Spain data had the highest accuracy.

4.2. SPMP techniques reported to be superior in comparative studies (RQ2)

From the results of MQ3, comparative studies about SPMP techniques presenting better performance were identified. Table 15 shows the details of these studies in terms of compared techniques and the results of the comparison; that is the techniques reported to be superior. The comparative studies proposed and/or evaluated SPMP techniques, and then compared them together or with other published studies such as: S2, S6, S9, S23, S26, S32, S37, and S38 (Table 15, second column).

As can be seen from Table 15 (third column), the MLP technique was reported superior in six studies, SVM was reported superior in four studies, GMDH, BN and ELM were reported to be superior in three studies, DT, MARS, BN, Neuro-GA, GEP, GA, and LSSVM techniques were reported to be superior in two studies each, and the rest of the techniques were reported only once.

Table 15. Summary of SPMP techniques reported to be superior

ID	Compared techniques	Techniques reported superior
S2	GRNN, WNN	GRNN
S6	BN, RT, BE, SS	BN
S9	MARS, MLR, ANN, SVR, RT	MARS
S10	GMM, SVM-RBF, DT	GMM
S15	AODE, SVM-LIN, NB, BN, KNN, C4.5, OneR, RBF	AODE
S19	PPR, ANN, MARS	PPR
S23	ANFIS, FFNN, FIS, RBF, GRNN	ANFIS
S26	ELM, RT, BE, SS, BN (S6)	ELM
S29	MLP, WNN, GRNN (S2)	MLP
S33	GMDH, GA, PNN, BN, RT, BE, SS (S6), MARS, MLR, ANN, RT, SVR (S9), GRNN, ANFIS (S23)	GMDH, GA, PNN
S35	MLP, WNN (S2)	MLP
S36	DT, LR, ANN	DT
S38	MLP, SVM, RBF, M5P	MLP, SVM
S41	DT, BPNN, SVM	BPNN
S42	MFL, ANFIS, SVM, PNN, RBF, BN (S6), MARS (S9)	MFL
S43	SBLLM, ELM, RT, BE, SS, BN (S6)	SBLLM, ELM
S44	K*, FSC, PR, KNN, MLR, LMSR, PPR, IR, RegByDisc, GPR, MLP, RBF, GRNN, GMDH, SVR, M5R, AR, ANFIS, DS, M5P, REPTree, LWL, CR, DTable, MARS (S9)	K*, FSC

Table 15 continued

ID	Compared techniques	Techniques reported superior
S45	XMC, KMC	XMC
S47	GMDH, GRNN, FF3LBPN	GMDH
S48	SS, BE, Oman & Hagemeister model [37]	SS, BE
S49	NB, BN, LgR, MLP	BN, MLP
S52	KN, MLR, BPNN, FFNN, GRNN	KN
	MLP, RBF, SVM, M5P	MLP, SVM
S54	MLP, RBF, SVM, DT	DT, RBF, SVM
	MLP, SVM, LgR, KMC, GEP	GEP, SVM
S55	Neuro-GA, ANN, BN, RT, BE, SS (S6),MARS, MLR, ANN, RT, SVR (S9)	Neuro-GA
S57	Neuro-GA, BN, RT, BE, SS (S6), MARS, MLR, ANN, RT, SVR (S9)	Neuro-GA
S58	FGA, AFGA, FPSO, MFPSO, FCSA, BN, RT, BE, SS (S6),MARS, MLR, ANN, RT, SVR (S9), FIS (S32), SVM-RBF (S37), MLP, RBF, SVM, M5P (S38)	FGA, AFGA, FPSO, MFPSO, FCSA
S59	GA, Dtable, RBF, BN, SMO	GA
S60	GGAL, GMDH, LR, M5R, DT, SVM, K*, JERN, BPNN, KN, PNN, GRNN	GGAL, GMDH
S61	SVM-SIG, SVM-RBF, SVM-LIN	SVM-SIG
S62	GEP, DFT, SVM, LR, MLP, RBF	GEP
S65	ELM-PLY, LR, NB, ELM-LIN, ELM-RBF, SVM-SIG, SVM-LIN, SVM-RBF	ELM-PLY
S66	Cubist, LR, Lasso, Elastic Net	Cubist
S68	M5P, MLR, MLP, SVR	M5P
S69	Neuro Fuzzy, BN, RT, BE, SS (S6), MARS, MLR, ANN, RT, SVR (S9), FL (S32), SVM, RBF (S37), MLP, RBF, SVM (S38), ANN, Neuro-GA (S55), Neuro-GA (S57)	Neuro Fuzzy
S70	SVM-RBF, SVM-LIN, SVM-SIG	SVM-RBF
S71	MARS, MLR, SVM	MARS
S72	LSSVM-LIN, LSSVM-RBF, LSSVM-SIG	LSSVM-LIN
S73	BN, MLP, LgR, NB, J48, NNge	BN, MLP
S77	LSSVM-RBF, LR, PR, LgR, DT, SVM-LIN, SVM-PLY, SVM-RBF, ELM-LIN, ELM-PLY, ELM-RBF, LSSVM-LIN, LSSVM-PLY, NGD, NGDM, NGDA, NNM	LSSVM-RBF

Table 16 provides a description of the techniques reported to be superior in more than two studies with their strengths and weaknesses as provided by the authors.

Some of the SPMP techniques identified in comparative studies have been reported to be superior in some studies and not in others. Note that Figure 15 includes techniques that were reported superior and not, at least one time each. For example, we note that:

- MLP technique was reported to be superior in six studies (S29, S35, S38, S54, S73), while not in six (S44, S54, S58, S62, S68, S69).
- SVM technique was reported superior in four studies (S38, S54, S61, S70) and not in eight (S10, S15, S42, S58, S60, S62, S65, S69).
- GMDH was reported superior in three studies (S33, S47, S60) and not in one (S44).
- MARS was reported superior in two studies (S9, S71) and not in eight (S19, S33, S42, S44, S55, S57, S58, S69).
- DT was reported superior in two studies (S36, S54) and not in four (S10, S41, S60, S77).
- BN was reported superior in three studies (S6, S49, S73) and not in eight (S15, S42, S43, S55, S57, S58, S59, S69).
- Neuro-GA was reported superior in two studies (S55, S57) and not in one (S69).
- RBF was reported superior in one study (S54) and not in 10 (S15, S23, S38, S42, S44, S54, S58, S59, S62, S69).
- M5P was reported superior in one study (S68) and not in four (S38, S44, S54, S58).
- GRNN was reported superior in one study (S2) and not in seven (S23, S29, S33, S44, S47, S52, S60).

Table 16. Strengths and weaknesses of the most accurate SPMP techniques

Technique	Description	Strength	Weakness
Multilayer perceptron (MLP)	"MLP are feed forward networks that consist of an input layer, one or more hidden layers of nonlinearly activating nodes and an output layer. Each node in one layer connects with a certain weight to every other node in the following layer" [38].	– "Minimizes the prediction error of the output variables" (S29, S35). – "Uses back propagation algorithm as the standard learning algorithm for any supervised learning" (S38, S54).	
Support vector machine (SVM)	"SVM are a group of supervised learning methods that can be applied to classification or regression problems" [39].	– "Minimizes the empirical error and maximizes the geometric margin" (S38, S54).	
Group method of data handling (GMDH)	"GMDH was introduced by Ivakhnenko and Ivakhnenko & Koppa for constructing an extremely high order regression type model and is based on forward multi-layer neural network structure where learning procedure is self-organized" [40, 41].	– "Ideal for complex, unstructured systems where the investigator is only interested in obtaining a high order input-output relationship" (S33). – "Predicts the outcome even with smaller training sets" (S33). – "Computational burden is reduced with GMDH" (S33). – "Can automatically filter out input properties that provide little information about location and shape of hyper surface" (S47).	– "Heuristic in nature and not based on a solid foundation as is regression analysis" (S33).

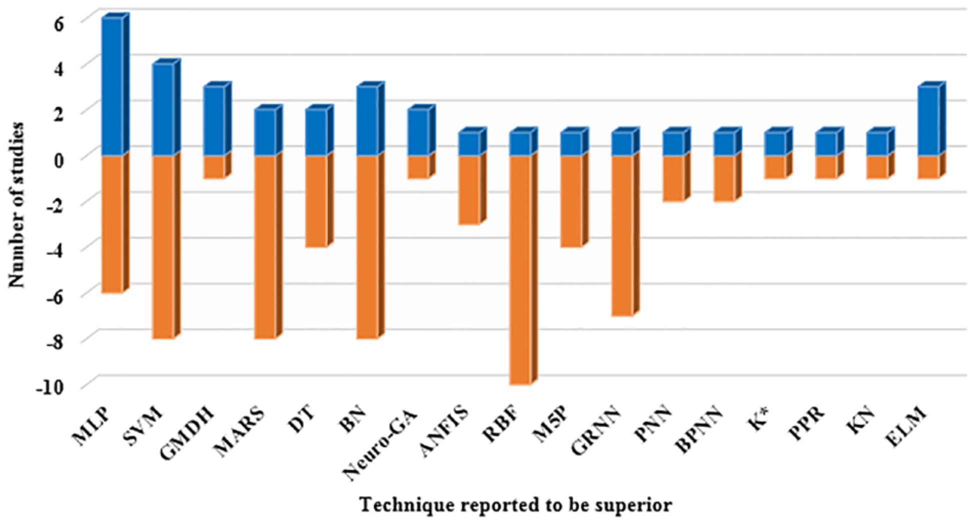


Figure 15. Techniques reported to be superior and not per study (bars above zero line indicate that techniques in horizontal axis are more accurate, whereas bars below zero line indicate that techniques in horizontal axis are not accurate)

– ELM was reported superior in three studies (S26, S43, S65) and not in one (S77).

From Figure 15, we note that no technique is definitively better than any other. Therefore, the choice of the best technique to predict maintainability is not obvious since every technique has advantages and drawbacks. Moreover, since the prediction context (e.g. dataset, accuracy criteria, etc.) is different among the studies, the literature results on the most accurate techniques are not sufficient to generalize the results.

4.3. Accuracy comparison of SPMP techniques reported to be superior in comparative studies (RQ3)

For a meaningful comparison, techniques are compared based on the same prediction context. From our investigation, we found that most of the comparative studies used the following prediction context:

– UIMS and QUES datasets (see MQ6),
– L&H and C&K metrics (see MQ7),
– Change dependent variable (see MQ7),
– MMRE and/or Pred(0.25), and/or Pred(0.30) accuracy criteria (RQ1), and
– Object-oriented software development paradigm (see MQ5).

The purpose of this section is to compare the techniques reported to be superior (see Table 15, third column), and which have this prediction context. Table 17 depicts the corresponding values of MMRE, and/or Pred(0.25), and/or Pred(0.30) for each technique per dataset. From comparative studies, 12 studies (20 experiments) were selected for UIMS and QUES datasets, and two studies (four experiments) for both datasets (i.e. the two datasets were merged). Note that one study may include more than one experiment.

Using MMRE and Pred as accuracy criteria for comparison, it is important to note that "to have a prediction model to be considered accurate, either MMRE < 0.25 and/or either Pred(0.25) >

Table 17. Prediction accuracy for UIMS, QUES, and BOTH datasets

ID	Technique	MMRE	Pred (0.25)	Pred (0.30)	Dataset	ID	Technique	MMRE	Pred (0.25)	Pred (0.30)	Dataset
S6	BN	0.97	0.44	0.46	UIMS	S6	BN	0.45	0.39	0.43	QUES
S9	MARS	1.86	0.28	0.28	UIMS	S9	MARS	0.32	0.48	0.59	QUES
S26	ELM	0.96	0.39	0.45	UIMS	S38	MLP	0.71	–	0.40	QUES
S38	MLP	1.39	–	0.23	UIMS	S38	SVM	0.44	–	0.51	QUES
S38	SVM	1.67	–	0.23	UIMS	S26	ELM	0.35	0.36	0.38	QUES
S42	MFL	0.53	0.30	0.35	UIMS	S42	MFL	0.27	0.52	0.62	QUES
S43	SBLLM	1.96	0.17	0.25	UIMS	S43	ELM	0.35	0.36	0.38	QUES
S43	ELM	0.96	0.17	0.25	UIMS	S43	SBLLM	0.34	0.50	0.56	QUES
S44	FSC	0.65	0.33	0.41	UIMS	S44	FSC	0.37	0.54	0.61	QUES
S44	K*	0.56	0.36	0.41	UIMS	S44	K*	0.27	0.56	0.66	QUES
S54	MLP	1.39	–	0.23	UIMS	S54	MLP	0.71	–	0.40	QUES
S54	SVM	1.64	–	0.23	UIMS	S54	SVM	0.44	–	0.56	QUES
S55	Neuro-GA	0.53	–	–	UIMS	S55	Neuro-GA	0.41	–	–	QUES
S57	Neuro-GA	0.31	–	–	UIMS	S57	Neuro-GA	0.37	–	–	QUES
S58	FGA	0.24	–	–	UIMS	S58	FGA	0.32	–	–	QUES
S58	AFGA	0.25	–	–	UIMS	S58	AFGA	0.32	–	–	QUES
S58	FPSO	0.27	–	–	UIMS	S58	FPSO	0.29	–	–	QUES
S58	MFPSO	0.25	–	–	UIMS	S58	MFPSO	0.32	–	–	QUES
S58	FCSA	0.27	–	–	UIMS	S58	FCSA	0.37	–	–	QUES
S69	Neuro-Fuzzy	0.28	–	–	UIMS	S69	Neuro-Fuzzy	0.33	–	–	QUES
S33	GMDH	0.21	0.69	0.72	BOTH	S33	PNN	0.23	0.68	0.75	BOTH
S33	GA	0.22	0.66	0.72	BOTH	S42	MFL	0.45	0.34	0.40	BOTH

0.75 or Pred(0.30) > 0.70, needed to be achieved" [35, 36]. That is, a low MMRE value or a high Pred(25) or Pred(30) value indicates good prediction accuracy. Table 17 shows that:

– For UIMS dataset, FGA, AFGA, and MFPSO achieved a significantly better prediction accuracy than the other techniques. They are near in terms of MMRE (MMRE = 0.24 for FGA, MMRE = 0.25 for AFGA and MFPSO). Besides, BN and ELM provide better accuracy than the other techniques in terms of Pred (Pred(0.25) = 0.44 and Pred(0.30) = 0.46 for BN followed by Pred(0.25) = 0.39 and Pred(0.30) = 0.45 for ELM).
– For QUES dataset, MFL and K* achieved the same MMRE value of 0.27. Moreover, they are near equal in terms of Pred: (Pred(0.25) = 0.52 and Pred(0.30) = 0.62 for MFL, while Pred(0.25) = 0.56 and Pred(0.30) = 0.66 for K*). Thus, the MFL and K* techniques provide better accuracy prediction compared to the remaining techniques.
– The GMDH, GA, and PNN techniques outperformed the MFL in both datasets (UIMS and QUES) with MMRE values of 0.21, 0.22 and 0.23, respectively, Pred(0.25) values of 0.69, 0.66 and 0.68, respectively, and Pred(0.30) values of 0.72, 0.72 and 0.75, respectively. Therefore, the GMDH was more accurate compared to the other techniques.

Here also, as stated in the previous section, no conclusion can be drawn about the most suitable technique for software product maintainability. Indeed, a technique can be more accurate in one study and less accurate in another. In addition, the accuracy of SPMP techniques is highly dependent on the prediction context (e.g. datasets used, accuracy criteria, etc.). Therefore, further studies are needed to reach a consensus on the most accurate technique for predicting maintainability of a software product.

5. Threats to validity

Three kinds of threats [10, 12] to the validity of this study are discussed as follows:

Construct validity: Construct threats to validity are related to the exhaustiveness and relevance of the primary studies. As previously noted, although maintainability and maintenance are different, they are often confounded and some studies do not make a clear distinction between them. Therefore, the search query was tailored to extract all available studies related to SPMP. Even though 82 primary studies were identified based on our search terms using keywords related to SPMP techniques, such a list may not be complete and a suitable study may have been left out. To ensure selection of the maximum number of studies, the search process was performed automatically on nine digital libraries and then manually by examining the reference section of the set of candidate studies to identify further studies. To identify additional studies, we established a set of inclusion and exclusion criteria.

Internal validity: Internal validity deals with data extraction and analysis. This threat is related to the reliability of the extracted data for the review, which can also be problematic. To accomplish this, two authors carried out the data extraction independently, keeping in mind the mapping and research questions, and their results compared. A third author reviewed the final results. When a disagreement arose, a discussion took place until an agreement was reached. If both authors extracted the same information for a specific paper, the extracted information was adopted. If the extracted information by the two authors was different for a specific paper, a meeting was held in which the full text of the paper was investigated.

External validity: External validity, which is very important for generalization of the results, is related to the context and conclusions drawn based on the data extracted. The results of this review were based only on the SPMP studies included in this paper. From each SPMP study, we extracted the dataset(s) used, and the dependent and independent variables validated empirically using experiments, surveys or case studies. Since we refrained from deriving or adjusting any data, the comparison between SPMP studies was impartial.

6. Conclusion and future guidelines

Industry and practitioners continue to search for effective ways to increase the maintainability of software products and reduce costs. In this paper, we reported on a follow-up systematic mapping and review to provide and summarize evidence on published empirical SPMP studies. After a thorough search of nine digital libraries and analysis of the relevance and quality of candidate studies, 82 primary studies were selected from 2000 to 2018. This study classified the SPMP studies according to publication year, publication source, research type, empirical approach, software application type, datasets, independent variables, dependent variables, and techniques used. The SPMP techniques were investigated from the following perspectives: prediction accuracy, techniques reported to be superior in comparative studies, and accuracy comparison. The main findings (Sections 3 and 4), how they differ from previous studies and new findings from this systematic mapping and review are summarized as follows:

– *What are the research types used in SPMP studies*? Empirical studies were broadly categorized into two categories: evaluation research and solution proposal. The most frequent SPMP studies were solution proposals, followed by evaluation research.
– *What empirical approaches were used*? The most frequently used empirical approach was history-based evaluation, followed by experiment and case study.
– *What datasets were used*? Historical datasets freely available to the public, such as those provided by software engineering researchers (SER) and private datasets, such as those used in academic or industrial (PSP) contexts were frequently used, followed by Software engineering researcher (SER) datasets.
– *What types of software applications were used*? Many types of software applications were used in these empirical studies, those used most frequently were object-oriented software applications.
– *What dependent and independent variables were used*?
  ◦ Maintainability in terms of the dependent variable to be predicted was most frequently expressed in terms of the number of changes made to the source code, followed by expert opinion based on an ordinal scale. This finding confirms, to some extent, the result of [4], but in reverse order, where it was reported that the most common dependent variable employed an ordinal scale based on expert judgment, followed by change measurements.
  ◦ For the independent variables (predictors), the most frequent predictors of software maintainability were those provided by Chidamber and Kemerer (C&K), Li and Henry (L&H), class diagram, source code size measures and McCabe complexity, which were gathered at the design and source code levels. This finding confirms, in reverse order, the result of [4]. Moreover, C&K and L&H measures, as predictors, were most often used to predict the maintainability expressed in terms of changes as a predicted output.
  ◦ The researchers used very few of the same data collection tools, thereby potentially leading to unknown error of measurement results since the measuring tools used have not been compared on similar benchmarks.
– *What techniques were used in SPMP*? The machine learning techniques were the most widely used in the literature. This finding is inconsistent with the results of [4] where the authors found that the commonly used maintainability prediction models were based on statistical techniques. This can be explained by the switch to machine learning techniques that have gained the interest of researchers since 2008.
– *What is the overall prediction accuracy of SPMP techniques*? Several accuracy criteria were used to evaluate SPMP techniques. MMRE and Pred accuracy criteria were the most frequently used in the selected primary studies. Based on these criteria, FNF was the most accurate technique for predicting maintainability expressed in terms of changes

based on MMRE and Pred(30), while ANN was the most accurate technique based on Pred(25).

- *Which SPMP techniques were reported superior in comparative studies*? We found that MLP, SVM, GMDH, and ELM were the most accurate techniques among selected comparative studies. Even if these techniques had better accuracy prediction in some studies, this was not the case in other studies. Therefore, no technique was definitively better than any other.
- *Which of the SPMP techniques reported to be superior in comparative studies also provided greater accuracy*? Accuracy comparison of techniques reported superior from comparative studies was carried out based on the same prediction context, in other words, the same datasets (UIMS or QUES or BOTH), the same metrics (L&H and C&K), the same dependent variable (Change), the same accuracy criteria (MMRE and/or Pred(0.25) and/or Pred(0.30), and the same software development paradigm (object-oriented). The results show that:
  - FGA, AFGA, and MFPSO achieved a significantly better prediction accuracy in terms of MMRE for the UIMS dataset,
  - MFL and K* were the most accurate for the QUES dataset, and
  - GMDH was the most accurate for both datasets.

  From this analysis we cannot conclude which is the most suitable technique for all cases as it is highly dependent on the prediction context (e.g. datasets used, accuracy criteria, etc.).

These findings may be useful to industry for comparing available SPMP models to improve the maintainability of software projects, and to researchers conducting further research into new SPMP techniques more performant than existing ones. Moreover, practitioners can choose the techniques used for predicting maintainability based on their prediction contexts as a solution in their practice.

In addition to the above findings, the following research gaps were identified:

- More free datasets should be made available to conduct empirical studies. In contrast to private datasets, public ones allow researchers to compare results in order to obtain generalizable results. Additional publicly available datasets can be used, such as the International Software Benchmarking Standard Group (ISBSG[1] repository of 8,261 completed software projects with more than 100 data fields, and PROMISE repository which is a collection of publicly available datasets grouped into one repository (http://promise.site.uottawa.ca/SERepository/).
- Most of the studies dealt with small datasets, such as UIMS and QUES with a single project each and related to projects developed using the Ada programming language. Large datasets based on the most frequently used programming languages in the industry are needed. This represents a serious challenge for the study of SPMP techniques. For instance, within the ISBSG, the most used programming languages include Java, COBOL, Oracle and .Net which represent 30%, 23%, 22% and 20%, respectively. It would be beneficial to SPMP research community to address this limitation.
- Moreover, dataset properties, such as type of data (categorical or numerical), missing values, outliers, etc., were not addressed by the research community.
- The majority of studies used data from OO software projects. As a result, there is a need for studies that examine maintainability for other types of applications such as web, mobile, model-driven, and cloud computing applications.
- SPMP studies are needed that focus on maintainability before delivery of the software product in order to detect problems and quality failures early, while the source code is not available. Such studies should be based on the 'requirements' and accordingly researchers must determine what 'independent variables' or 'predictors' can be collected based on the requirements.
- Few studies address maintainability from the process level. More studies are needed to in-

[1]ISBSG, Development and Enhancement repository, February 2018, (http://www.isbsg.org).

vestigate how software development factors as well as software process management factors (such as project planning, requirement analysis, architectural design, development team, etc.) affect software maintainability.
– Few studies use ensemble techniques. More studies are needed using ensemble techniques since they use various single techniques to obtain a more accurate result.

Researchers interested in carrying out future research on SPMP, including empirical studies and benchmarking studies, would do well to investigate these research gaps and suggested research avenues.

References

[1] P. Bourque and R.E. Fairley, *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0.* IEEE Computer Society Press, 2014.

[2] P. Oman and J. Hagemeister, "Construction and testing of polynomials predicting software maintainability," *Journal of Systems and Software*, Vol. 24, No. 3, 1994, pp. 251–266.

[3] A. Kaur and K. Kaur, "Statistical comparison of modelling methods for software maintainability prediction," *International Journal of Software Engineering and Knowledge Engineering*, Vol. 23, No. 6, 2013, pp. 743–774.

[4] M. Riaz, E. Mendes, and E. Tempero, "A systematic review of software maintainability prediction and metrics," in *Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement.* IEEE Computer Society, 2009, pp. 367–377.

[5] M. Riaz, "Maintainability prediction of relational database-driven applications: A systematic review," in *16th International Conference on Evaluation & Assessment in Software Engineering.* IET, 2012, pp. 263–272.

[6] B.A. Orenyi, S. Basri, and L.T. Jung, "Object-oriented software maintainability measurement in the past decade," in *International Conference on Advanced Computer Science Applications and Technologies (ACSAT).* IEEE, 2012, pp. 257–262.

[7] S.K. Dubey, A. Sharma, and A. Rana, "Analysis of maintainability models for object oriented system," *International Journal on Computer Science and Engineering*, Vol. 3, No. 12, 2011, p. 3837.

[8] A.M. Fernández-Sáez, M. Genero, and M.R. Chaudron, "Empirical studies concerning the maintenance of UML diagrams and their use in the maintenance of code: A systematic mapping study," *Information and Software Technology*, Vol. 55, No. 7, 2013, pp. 1119–1142.

[9] A.M. Fernández-Sáez, D. Caivano, M. Genero, and M.R. Chaudron, "On the use of UML documentation in software maintenance: Results from a survey in industry," in *18th International Conference on Model Driven Engineering Languages and Systems (MODELS).* IEEE, 2015, pp. 292–301.

[10] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Information and Software Technology*, Vol. 64, 2015, pp. 1–18.

[11] B.A. Kitchenham, D. Budgen, and O.P. Brereton, "Using mapping studies as the basis for further research – A participant-observer case study," *Information and Software Technology*, Vol. 53, No. 6, 2011, pp. 638–651.

[12] A. Idri, F.A. Amazal, and A. Abran, "Analogy-based software development effort estimation: A systematic mapping and review," *Information and Software Technology*, Vol. 58, 2015, pp. 206–230.

[13] W. Li and H. Sallie, "Object oriented metrics that predict maintainability," *Journal of Systems and Software*, Vol. 23, No. 2, 1993, pp. 111–122.

[14] A. Kaur, K. Kaur, and K. Pathak, "Software maintainability prediction by data mining of software code metrics," in *International Conference on Data Mining and*

Intelligent Computing (ICDMIC). IEEE, 2014, pp. 1–6.

[15] P. Omidyar, *eBay web services*. [Online]. http://developer.ebay.com [Accessed: 2018-10-26].

[16] B.R. Reddy and O. Aparajita, "Performance of maintainability index prediction models: A feature selection based study," *Evolving Systems*, 2017.

[17] A. Jain, S. Tarwani, and A. Chug, "An empirical investigation of evolutionary algorithm for software maintainability prediction," in *Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, 2016, pp. 1–6.

[18] M. Genero, J.A. Olivas, M. Piattini, and F.P. Romero, "A controlled experiment for corroborating the usefulness of class diagram metrics at the early phases of OO developments," in *ADIS*, 2001.

[19] X. Jin, Y. Liu, J. Ren, A. Xu, and R. Bie, "Locality preserving projection on source code metrics for improved software maintainability," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2006, pp. 877–886.

[20] K. Beck and E. Gamma, *JUnit*. [Online]. https://junit.org [Accessed: 2018-10-26].

[21] R. Malhotra and A. Chug, "Application of group method of data handling model for software maintainability prediction using object oriented systems," *International Journal of System Assurance Engineering and Management*, Vol. 5, No. 2, 2014, pp. 165–173.

[22] S.J. Sayyad and T. Menzies, *The PROMISE Repository of Software Engineering Databases*, School of Information Technology and Engineering, University of Ottawa, Canada. [Online]. http://promise.site.uottawa.ca/SERepository Accessed: 2017-09-11.

[23] *Software Engineering – Product Quality – Part 2: External Metrics, Part 3: Internal Metrics, Part 4: Quality in Use Metrics*, ISO/IEC Std. TR 9126-2-3-4, 2003, 2004.

[24] *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models*, Geneva, ISO Std. 25 010, 2010.

[25] D. Spinellis, *Chidamber and Kemerer Java Metrics (CKJM)*. [Online]. http://www.spinellis.gr/sw/ckjm/ [Accessed: 2017-11-21].

[26] *Jetbrains Homepage*. [Online]. http://www.jetbrains.com/idea/ [Accessed: 2017-11-19].

[27] *Krakatau Professional Homepage*. [Online]. http://www.powersoftware.com/kp/ [Accessed: 2017-11-19].

[28] R. Ferenc, A. Beszedes, M. Tarkiainen, and T. Gyimothy, "Columbus – reverse engineering tool and schema for C++," in *Proceedings of the International Conference on Software Maintenance (ICSM '02)*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 172–181.

[29] G.A. Di Lucca, A.R. Fasolino, F. Pace, P. Tramontana, and U. De Carlini, "WARE: A tool for the reverse engineering of web applications," in *Proceedings of the Sixth European Conference on Software Maintenance and Reengineering*, 2002, pp. 241–250.

[30] B.R. Reddy, S. Khurana, and A. Ojha, "Software maintainability estimation made easy: A comprehensive tool coin," in *Proceedings of the Sixth International Conference on Computer and Communication Technology 2015*. New York: ACM, 2015, pp. 68–72.

[31] *Analyst4j standard tool*. [Online]. https://codeswat.com/ [Accessed: 2018-01-01].

[32] S. Almugrin, W. Albattah, and A. Melton, "Using indirect coupling metrics to predict package maintainability and testability," *Journal of System and Software*, Vol. 121, No. C, 2016, pp. 298–310.

[33] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," *Applied Soft Computing*, Vol. 27, 2015, pp. 504–518.

[34] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," *Information and Software Technology*, Vol. 54, No. 1, 2012, pp. 41–59.

[35] S.D. Conte, H.E. Dunsmore, and Y. Shen, *Software engineering metrics and models*. Benjamin-Cummings Publishing Co., Inc., 1986.

[36] A.R. Gray and S.G. MacDonell, "A comparison of techniques for developing predictive models of software metrics," *Information and software technology*, Vol. 39, No. 6, 1997, pp. 425–437.

[37] P. Oman and J. Hagemeister, "Construction and testing of polynomials predicting software maintainability," *Journal of Systems and Software*, Vol. 24, No. 3, 1994, pp. 251–266.

[38] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[39] V. Vapnik, *The nature of statistical learning theory*. New York: Springer-Verlag, 1995.

[40] A.G. Ivakhnenko, "The group method of data of handling; A rival of the method of stochastic approximation," *Soviet Automatic Control*, Vol. 13, 1968, pp. 43–55.

[41] A.G. Ivakhnenko and Y. Koppa, "Regularization of decision functions in the group method of data handling," *Soviet Automatic Control*, Vol. 15, No. 2, 1970, pp. 28–37.

[42] P. Brereton, B.A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," *Journal of Systems and Software*, Vol. 80, No. 4, 2007, pp. 571–583.

[43] J. Magne and M. Shepperd, "A systematic review of software development cost estimation studies," *IEEE Transactions on Software Engineering*, Vol. 33, No. 1, 2007, pp. 33–53.

[44] S. Muthanna, K. Kontogiannis, K. Ponnambalam, and B. Stacey, "A maintainability model for industrial software systems using design level metrics," in *Proceedings Seventh Working Conference on Reverse Engineering*. IEEE, 2000, pp. 248–256.

[45] M. Thwin and T. Quah, "Application of neural networks for estimating software maintainability using object-oriented metrics," in *International Conference on Software Engineering and Knowledge Engineering*, 2003, pp. 69–73.

[46] M. Genero, M. Piattini, E. Manso, and G. Cantone, "Building UML class diagram maintainability prediction models based on early metrics," in *5th International Workshop on Enterprise Networking and Computing in Healthcare Industry*. IEEE, 2003, pp. 263–275.

[47] M. Kiewkanya, N. Jindasawat, and P. Muenchaisri, "A methodology for constructing maintainability model of object-oriented design," in *Fourth International Conference on Quality Software*. IEEE, 2004, pp. 206–213.

[48] A.R. Di Lucca, Giuseppe A and Fasolino, P. Tramontana, and C.A. Visaggio, "Towards the definition of a maintainability model for web applications," in *Eighth European Conference on Software Maintenance and Reengineering*. IEEE, 2004, pp. 279–287.

[49] C. Van Koten and A. Gray, "An application of Bayesian network for predicting object-oriented software maintainability," *Information and Software Technology*, Vol. 48, No. 1, 2006, pp. 59–67.

[50] J.H. Hayes and L. Zhao, "Maintainability prediction: A regression analysis of measures of evolving systems," in *21st International Conference on Software Maintenance (ICSM '05)*. IEEE, 2005, pp. 601–604.

[51] S.C. Misra, "Modeling design/coding factors that drive maintainability of software systems," *Software Quality Journal*, Vol. 13, No. 3, 2005, pp. 297–320.

[52] Y. Zhou and H. Leung, "Predicting object-oriented software maintainability using multivariate adaptive regression splines," *Journal of Systems and Software*, Vol. 80, No. 8, 2007, pp. 1349–1361.

[53] S.S. Dahiya, J.K. Chhabra, and S. Kumar, "Use of genetic algorithm for software maintainability metrics' conditioning," in

15th International Conference on Advanced Computing and Communications. IEEE, 2007, pp. 87–92.

[54] M. Genero, E. Manso, A. Visaggio, G. Canfora, and M. Piattini, "Building measure-based prediction models for UML class diagram maintainability," *Empirical Software Engineering*, Vol. 12, No. 5, 2007, pp. 517–549.

[55] K. Shibata, K. Rinsaka, T. Dohi, and H. Okamura, "Quantifying software maintainability based on a fault-detection/correction model," in *13th Pacific Rim International Symposium on Dependable Computing*. IEEE, 2007, pp. 35–42.

[56] K. Aggarwal, Y. Singh, A. Kaur, and R. Malhotra, "Application of artificial neural network for predicting maintainability using object-oriented metrics," *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, Vol. 2, No. 10, 2008, pp. 3552–3556.

[57] Y. Tian, C. Chen, and C. Zhang, "AODE for source code metrics for improved software maintainability," in *Fourth International Conference on Semantics, Knowledge and Grid*. IEEE, 2008, pp. 330–335.

[58] Y. Zhou and B. Xu, "Predicting the maintainability of open source software using design metrics," *Wuhan University Journal of Natural Sciences*, Vol. 13, No. 1, 2008, pp. 14–20.

[59] H. Yu, G. Peng, and W. Liu, "An application of case based reasoning to predict structure maintainability," in *International Conference on Computational Intelligence and Software Engineering*. IEEE, 2009, pp. 1–5.

[60] A. Sharma, P. Grover, and R. Kumar, "Predicting maintainability of component-based systems by using fuzzy logic," in *International Conference on Contemporary Computing*. Springer, 2009, pp. 581–591.

[61] L. Wang, X. Hu, Z. Ning, and W. Ke, "Predicting object-oriented software maintainability using projection pursuit regression," in *First International Conference on Information Science and Engineering*. IEEE, 2009, pp. 3827–3830.

[62] H. Mittal and P. Bhatia, "Software maintainability assessment based on fuzzy logic technique," *ACM SIGSOFT Software Engineering Notes*, Vol. 34, No. 3, 2009, pp. 1–5.

[63] M.O. Elish and K.O. Elish, "Application of treenet in predicting object-oriented software maintainability: A comparative study," in *13th European Conference on Software Maintenance and Reengineering*. IEEE, 2009, pp. 69–78.

[64] S. Rizvi and R.A. Khan, "Maintainability estimation model for object-oriented software in design phase (MEMOOD)," *arXiv preprint arXiv:1004.4447*, 2010.

[65] A. Kaur, K. Kaur, and R. Malhotra, "Soft computing approaches for prediction of software maintenance effort," *International Journal of Computer Applications*, Vol. 1, No. 16, 2010, pp. 69–75.

[66] C. Jin and J.A. Liu, "Applications of support vector mathine and unsupervised learning for predicting maintainability using object-oriented metrics," in *Second International Conference on Multimedia and Information Technology*, Vol. 1. IEEE, 2010, pp. 24–27.

[67] L. Cai, Z. Liu, J. Zhang, W. Tong, and G. Yang, "Evaluating software maintainability using fuzzy entropy theory," in *9th International Conference on Computer and Information Science*. IEEE, 2010, pp. 737–742.

[68] S.O. Olatunji, Z. Rasheed, K. Sattar, A. Al-Mana, M. Alshayeb, and E. El-Sebakhy, "Extreme learning machine as maintainability prediction model for object-oriented software systems," *Journal of Computing*, Vol. 2, No. 8, 2010, pp. 49–56.

[69] P. Dhankhar, H. Mittal, and A. Mittal, "Maintainability prediction for object oriented software," *International Journal of Advances in Engineering Sciences*, Vol. 1, No. 1, 2011, pp. 8–11.

[70] S.K. Dubey and A. Rana, "A fuzzy approach for evaluation of maintainability of object oriented software system," *International Journal of Computer Applications*, Vol. 49, No. 21, 2012.

[71] S.K. Dubey, A. Rana, and Y. Dash, "Maintainability prediction of objec oriented software system by multilayer perceptron model," *ACM SIGSOFT Software Engineering Notes*, Vol. 37, No. 5, 2012, pp. 1–4.

[72] N. Tagoug, "Maintainability assessment in object-oriented system design," in *International Conference on Information Technology and e-Services*. IEEE, 2012, pp. 1–5.

[73] S. Sharawat, "Software maintainability prediction using neural networks," *environment*, Vol. 3, No. 5, 2012, pp. 750–755.

[74] M. Al-Jamimi, Hamdi A and Ahmed, "Prediction of software maintainability using fuzzy logic," in *International Conference on Computer Science and Automation Engineering*. IEEE, 2012, pp. 702–705.

[75] R. Malhotra and A. Chug, "Software maintainability prediction using machine learning algorithms," *Software Engineering: An International Journal*, Vol. 2, No. 2, 2012.

[76] T. Bakota, P. Hegedűs, G. Ladányi, P. Körtvélyesi, R. Ferenc, and T. Gyimóthy, "A cost model based on software maintainability," in *28th International Conference on Software Maintenance (ICSM)*. IEEE, 2012, pp. 316–325.

[77] Y. Dash, S.K. Dubey, and A. Rana, "Maintainability prediction of object oriented software system by using artificial neural network approach," *International Journal of Soft Computing and Engineering (IJSCE)*, Vol. 2, No. 2, 2012, pp. 420–423.

[78] P. Hegedűs, G. Ladányi, I. Siket, and R. Ferenc, "Towards building method level maintainability models based on expert evaluations," in *Computer Applications for Software Engineering, Disaster Recovery, and Business Continuity*. Springer, 2012, pp. 146–154.

[79] D. Chandra, "Support vector approach by using radial kernel function for prediction of software maintenance effort on the basis of multivariate approach," *International Journal of Computer Applications*, Vol. 51, No. 4, 2012.

[80] H. Aljamaan, M.O. Elish, and I. Ahmad, "An ensemble of computational intelligence models for software maintenance effort prediction," in *International Work-Conference on Artificial Neural Networks*. Springer, 2013, pp. 592–603.

[81] P. Hegedűs, T. Bakota, G. Ladányi, C. Faragó, and R. Ferenc, "A drill-down approach for measuring maintainability at source code element level," *Electronic Communications of the EASST*, Vol. 60, 2013.

[82] X.L. Hao, X.D. Zhu, and L. Liu, "Research on software maintainability evaluation based on fuzzy integral," in *International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE)*. IEEE, 2013, pp. 1279–1282.

[83] F. Ye, X. Zhu, and Y. Wang, "A new software maintainability evaluation model based on multiple classifiers combination," in *International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE)*. IEEE, 2013, pp. 1588–1591.

[84] H.A. Ahmed, Moataz A and Al-Jamimi, "Machine learning approaches for predicting software maintainability: A fuzzy-based transparent model," *IET software*, Vol. 7, No. 6, 2013, pp. 317–326.

[85] S.O. Olatunji and A. Ajasin, "Sensitivity-based linear learning method and extreme learning machines compared for software maintainability prediction of object-oriented software systems," *ICTACT Journal On Soft Computing*, Vol. 3, No. 03, 2013.

[86] A. Mehra and S.K. Dubey, "Maintainability evaluation of object-oriented software system using clustering techniques," *Internationa Journal of Computers and Technology*, Vol. 5, No. 02, 2013, pp. 136–143.

[87] J. Al Dallal, "Object-oriented class maintainability prediction using internal qual-

ity attributes," *Information and Software Technology*, Vol. 55, No. 11, 2013, pp. 2028–2048.

[88] A. Kaur, K. Kaur, and K. Pathak, "A proposed new model for maintainability index of open source software," in *Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization.* IEEE, 2014, pp. 1–6.

[89] A. Pratap, R. Chaudhary, and K. Yadav, "Estimation of software maintainability using fuzzy logic technique," in *International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT).* IEEE, 2014, pp. 486–492.

[90] L. Geeta, A. Kavita, and B. Rizwan, "Maintainability measurement model for object oriented design," *International Journal of Advanced Research in Computer Science and Software Engineering*, Vol. 4, No. 11, 2014, pp. 945–956.

[91] R. Malhotra and A. Chug, "A metric suite for predicting software maintainability in data intensive applications," in *Transactions on Engineering Technologies.* Springer Netherlands, 2014, pp. 161–175.

[92] S. Misra and F. Egoeze, "Framework for maintainability measurement of web application for efficient knowledge-sharing on campus intranet," in *Computational Science and Its Applications – ICCSA 2014.* Cham: Springer International Publishing, 2014, pp. 649–662.

[93] M.O. Elish, H. Aljamaan, and I. Ahmad, "Three empirical studies on predicting software maintainability using ensemble methods," *Soft Computing*, Vol. 19, No. 9, 2015, pp. 2511–2524.

[94] L. Kumar and S.K. Rath, "Neuro-genetic approach for predicting maintainability using Chidamber and Kemerer software metrics suite," in *Recent Advances in Information and Communication Technology 2015.* Cham: Springer International Publishing, 2015, pp. 31–40.

[95] S.O. Olatunji and A. Selamat, "Type-2 fuzzy logic based prediction model of object oriented software maintainability," in *Intelligent Software Methodologies, Tools and Techniques.* Cham: Springer International Publishing, 2015, pp. 329–342.

[96] L. Kumar, D.K. Naik, and S.K. Rath, "Validating the effectiveness of object-oriented metrics for predicting maintainability," *Procedia Computer Science*, Vol. 57, 2015, pp. 798–806.

[97] L. Kumar and S.K. Rath, "Hybrid functional link artificial neural network approach for predicting maintainability of object-oriented software," *Journal of Systems and Software*, Vol. 121, No. C, 2016, pp. 170–190.

[98] A. Chug and R. Malhotra, "Benchmarking framework for maintainability prediction of open source software using object oriented metrics," *International Journal of Innovative Computing, Information and Control*, Vol. 12, No. 2, 2016, pp. 615–634.

[99] L. Kumar, K. Mukesh, and K.R. Santanu, "Maintainability prediction of web service using support vector machine with various kernel methods," *International Journal of System Assurance Engineering and Management*, Vol. 8, No. 2, 2017, pp. 205–6222.

[100] S. Tarwani and A. Chug, "Predicting maintainability of open source software using gene expression programming and bad smells," in *5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 2016, pp. 452–459.

[101] S. Tarwani and A. Chug, "Sequencing of refactoring techniques by greedy algorithm for maximizing maintainability," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2016, pp. 1397–1403.

[102] L. Kumar, S.K. Rath, and A. Sureka, "Empirical analysis on effectiveness of source code metrics for predicting change-proneness," in *Proceedings of the 10th Innovations in Software Engineering Conference*, ISEC '17. New York, NY, USA: ACM, 2017, pp. 4–14.

[103] G. Kanika and C. Anuradha, "Evaluation of instance-based feature subset selection

algorithm for maintainability prediction," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, pp. 1482–1487.

[104] K. Shivani and T. Kirti, "Maintainability assessment for software by using a hybrid fuzzy multi-criteria analysis approach," *Management Science Letters*, Vol. 7, 2017, pp. 255–274.

[105] K. Lov and K.R. Santanu, "Software maintainability prediction using hybrid neural network and fuzzy logic approach with parallel computing concept," *International Journal of System Assurance Engineering and Management*, Vol. 8, No. S2, 2017, pp. 1487–1502.

[106] L. Kumar, A. Krishna, and S.K. Rath, "The impact of feature selection on maintainability prediction of service-oriented applications," *Service Oriented Computing and Applications*, Vol. 11, No. 2, 2017, pp. 137–161.

[107] L. Kumar, S.K. Rath, and A. Sureka, "Using source code metrics and multivariate adaptive regression splines to predict maintainability of service oriented software," in *18th International Symposium on High Assurance Systems Engineering (HASE)*, 2017, pp. 88–95.

[108] L. Kumar, S.K. Rath, and A. Sureka, "Using source code metrics to predict change-prone web services: A case-study on ebay services," in *IEEE Workshop on Machine Learning Techniques for Software Quality Evaluation – MaLTeSQuE*. IEEE, 2017, pp. 1–7.

[109] R. Malhotra and R. Jangra, "Prediction and assessment of change prone classes using statistical and machine learning techniques," *Journal of Information Processing Systems*, Vol. 13, No. 4, 2017, pp. 778–804.

[110] G. Szőke, G. Antal, C. Nagy, R. Ferenc, and T. Gyimóthy, "Empirical study on refactoring large-scale industrial systems and its effects on maintainability," *Journal of Systems and Software*, Vol. 129, 2017, pp. 107–126.

[111] Y. Gokul and M. Gopal, "An authoritative method using fuzzy logic to evaluate maintainability index and utilizability of software," *Advances in Modelling and Analysis B*, Vol. 60, No. 3, 2017, pp. 566–580.

[112] P. Hegedűs, I. Kádár, R. Ferenc, and T. Gyimóthy, "Empirical evaluation of software maintainability based on a manually validated refactoring dataset," *Information and Software Technology*, Vol. 95, No. 1, 2018, pp. 313–327.

[113] L. Kumar and S. Ashish, "A comparative study of different source code metrics and machine learning algorithms for predicting change proneness of object oriented systems," *arXiv preprint arXiv:1712.07944*, 2018.

[114] G. Scanniello, C. Gravino, M. Genero, J.A. Cruz-Lemus, and G. Tortora, "On the impact of UML analysis models on source-code comprehensibility and modifiability," *ACM Trans. Softw. Eng. Methodol.*, Vol. 23, No. 2, 2014, pp. 13:1–13:26.

[115] A.M. Fernández-Sáez, M.R.V. Chaudron, M. Genero, and I. Ramos, "Are forward designed or reverse-engineered UML diagrams more helpful for code maintenance?: A controlled experiment," in *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*. New York, NY, USA: ACM, 2013, pp. 60–71.

[116] G. Scanniello, C. Gravino, G. Tortora, M. Genero, M. Risi, J.A. Cruz-Lemus, and G. Dodero, "Studying the effect of UML-based models on source-code comprehensibility: Results from a long-term investigation," in *Proceedings of the 16th International Conference on Product-Focused Software Process Improvement*, Vol. 9459, New York, 2015, pp. 311–327.

[117] A.M. Fernández-Sáez, M. Genero, D. Caivano, and M.R. Chaudron, "Does the level of detail of UML diagrams affect the maintainability of source code?: A family of experiments," *Empirical Software Engineering*, Vol. 21, No. 1, 2016, pp. 212–259.

[118] G. Scanniello, C. Gravino, M. Genero, J.A. Cruz-Lemus, G. Tortora, M. Risi, and G. Dodero, "Do software models based

on the UML aid in source-code comprehensibility? Aggregating evidence from 12 controlled experiments," *Empirical Software Engineering*, Vol. 23, No. 5, 2018, pp. 2695–2733.

Appendix A

Table A1. Research approaches

Research approaches	What is it [42]
ER	Empirical studies that evaluate and/or compare existing SPMP techniques.
SP	Empirical studies in which an SPMP technique is proposed, either as a new technique or as a significant adaptation of an existing one, or propose a solution to a defined problem.

Table A2. Empirical types

Empirical types	What is it [43]
HbE	Studies evaluating SPMP techniques of previously completed software projects.
Ex	An empirical method applied under controlled conditions to evaluate an SPMP technique.
CS	An empirical study that investigates an SPMP technique in a real-life context, e.g. in-depth study of the prediction processes of one, or a very small number, of software projects.

Table A3. QA score of selected primary studies

ID	Author	QA1	QA2	QA3	QA4	QA5	QA6	QA7	Score
S1	S. Muthanna et al.	1	1	1	1	1	0	1	6
S2	M.M.T Thwin et al.	1	1	1	1	0.5	1	1	6.6
S3	M. Genero et al.	1	1	1	1	0.5	1	1	6.6
S4	M. Kiewkayna et al.	1	1	0.5	1	0.5	0	1	5
S5	G.A.D. Lucca et al.	1	1	0.5	1	0.5	0.5	1	5.5
S6	C.V. Koten et al.	1	1	1	1	1	1	1	7
S7	J.H. Hayes et al.	1	0.5	1	1	0.5	0.5	1	5.5
S8	S.C. Misra	1	1	1	1	0.5	1	1	6.6
S9	Y. Zhou et al.	1	1	1	1	1	1	1	7
S10	X. Jin et al.	1	1	1	1	1	1	1	7
S11	S.S. Dahiya et al.	1	0.5	0.5	1	0.5	0.5	1	5
S12	M. Genero et al.	1	1	0.5	1	1	0.5	1	6
S13	K. Shibata et al.	1	0.5	0.5	1	0.5	0.5	1	5
S14	K.K.Aggarwal et al.	0.5	1	0.5	0.5	0.5	1	0.5	4.5
S15	Y. Thian et al.	1	1	1	1	1	1	1	7
S16	Y. Zhou et al.	1	1	1	1	1	1	1	7
S17	H. Yu et al.	1	1	1	1	1	0	1	6
S18	A. Sharma et al.	1	1	1	1	0.5	0.5	1	6
S19	W. Li-jin et al.	1	1	1	1	1	1	1	7
S20	H. Mittal et al.	1	1	0.5	1	0.5	1	1	6

Table A3 continued

ID	Author	QA1	QA2	QA3	QA4	QA5	QA6	QA7	Score
S21	M. O. Elish et al.	1	1	1	1	1	1	1	7
S22	S. Rizvi et al.	1	1	1	1	0.5	1	1	6.6
S23	A.Kaur et al.	1	1	1	1	1	1	1	7
S24	C. Jin et al.	1	1	1	0.5	1	1	1	6.6
S25	L. CAI et al.	1	0.5	0.5	1	0.5	0.5	1	5
S26	S. O. Olatunji et al.	1	1	1	1	1	1	1	7
S27	P. Dhankhar et al.	1	1	0.5	1	0.5	0.5	1	5.5
S28	S.K. Dubey et al.	1	1	0.5	1	0.5	0.5	1	5.5
S29	S. K. Dubey et al.	1	1	1	1	1	1	1	7
S30	N. Tagoug et al.	1	0.5	1	1	1	0.5	1	5.5
S31	S. Sharawat et al.	1	1	1	1	0.5	0.5	1	6
S32	H.A. Al-Jamimi et al.	1	1	1	1	1	1	1	7
S33	R. Malhotra et al.	1	1	1	1	1	1	1	7
S34	T. Bakota et al.	1	1	1	1	1	1	1	7
S35	Y. Dash et al.	1	1	1	1	1	1	1	7
S36	P. Hegedűs et al.	1	1	1	1	1	1	1	7
S37	D. Chandra	1	1	1	1	1	1	1	7
S38	H. Aljamaan et al.	1	1	1	1	1	1	1	7
S39	P. Hegedűs et al.	1	1	0.5	1	1	1	1	6.6
S40	X.L. Hao et al.	1	1	1	1	1	0	1	6
S41	F. Ye et al.	1	1	1	1	0.5	1	1	6.6
S42	M.A. Ahmed et al.	1	1	1	1	1	1	1	7
S43	S.O. Olatunji et al.	1	1	1	1	1	1	1	7
S44	A. Kaur et al.	1	1	1	1	1	1	1	7
S45	A. Mehra et al.	1	1	1	1	0.5	1	1	6.6
S46	J. Al Dallal.	1	1	1	1	1	1	1	7
S47	R. Malhotra et al.	1	1	1	1	1	1	1	7
S48	A. Kaur et al.	1	1	1	1	1	1	1	7
S49	A. Kaur et al.	1	1	1	1	1	1	1	7
S50	A. Pratap et al.	1	1	0.5	1	1	0.5	0.5	5.5
S51	R. Kumar et al.	1	1	1	1	0.5	1	1	6.6
S52	R. Malhotra et al.	1	1	1	1	1	1	1	7
S53	S. Misra et al.	1	1	0.5	1	0.5	0.5	0.5	5
S54	M.O. Elish et al.	1	1	1	1	1	1	1	7
S55	L. Kumar et al.	1	1	1	1	1	1	1	7
S56	S.O. Olatunji et al.	1	1	1	1	1	1	1	7
S57	A.K. Soni et al.	1	1	1	1	1	1	1	7
S58	L. Kumar et al.	1	1	1	1	1	1	1	7
S59	A. Jain et al.	1	1	1	1	1	1	1	7
S60	A. Chug et al.	1	1	1	1	1	1	1	7
S61	L. Kumar et al.	1	1	1	1	1	1	1	7
S62	S. Tarwani et al.	1	1	1	1	1	1	1	7
S63	S. Almugrin et al.	1	1	1	1	1	0.5	1	6.5
S64	S. Tarwani et al.	1	1	1	1	0.5	0.5	1	6
S65	L. Kumar et al.	1	1	1	1	1	1	1	7
S66	K. Gupta et al.	1	1	1	1	1	1	1	7
S67	S. Kundu et al.	1	1	0.5	1	1	0	1	5.5
S68	B.R. Reddy et al.	1	1	1	1	1	1	1	7
S69	L. Kumar et al.	1	1	1	1	1	1	1	7
S70	L. Kumar et al.	1	1	1	1	1	1	1	7
S71	L. Kumar et al.	1	1	1	1	1	1	1	7

Table A3 continued

ID	Author	QA1	QA2	QA3	QA4	QA5	QA6	QA7	Score
S72	L. Kumar et al.	1	1	1	1	1	1	1	7
S73	R. Malhotra et al.	1	1	1	1	1	1	1	7
S74	G. Szoke et al.	1	1	1	1	0.5	0	1	5.5
S75	Y. Gokul et al.	1	1	0.5	1	0.5	0	1	5
S76	P. Hegedűs et al.	1	1	1	1	1	0	1	6
S77	L. Kumar et al.	1	1	1	1	1	1	1	7
S78	G. Scanniello et al.	1	1	1	0.5	0.5	0	1	5
S79	A.M. Fernández-Sáez et al.	1	1	1	0.5	0.5	0	1	5
S80	G. Scanniello et al.	1	1	1	0.5	0.5	0	1	5
S81	A.M. Fernández-Sáez et al.	1	1	1	0.5	0.5	0	1	5
S82	G. Scanniello et al.	1	1	1	0.5	0.5	0	1	5

Table A4. Search results for each of the nine databases

Database name	# of search results	# of duplicate studies	# of candidate studies	# of relevant studies
IEEE Explore	1678	15	100	28
Science Direct	5938	20	30	9
Springer Link	8715	45	71	18
Ebsco	1601	16	6	1
ACM Digital Library	530	14	10	5
Google Scholar	22090	30	77	10
dblp	120	80	20	2
Scopus	270	17	23	0
Jstore	399	26	4	2
Total	41341	263	341	75

Table A5. List of the 82 selected studies

ID	Author	Ref.	Title
S1	S. Muthanna et al.	[44]	S. Muthanna, K. Kontogiannis, K. Ponnambalam, and B. Stacey, "A maintainability model for industrial software systems using design level metrics," in *Proceedings Seventh Working Conference on Reverse Engineering.* IEEE, 2000, pp. 248–256
S2	M.M.T Thwin et al.	[45]	M. Thwin and T. Quah, "Application of neural networks for estimating software maintainability using object-oriented metrics," in *International Conference on Software Engineering and Knowledge Engineering*, 2003, pp. 69–73
S3	M. Genero et al.	[46]	M. Genero, M. Piattini, E. Manso, and G. Cantone, "Building UML class diagram maintainability prediction models based on early metrics," in *5th International Workshop on Enterprise Networking and Computing in Healthcare Industry.* IEEE, 2003, pp. 263–275
S4	M. Kiewkayna et al.	[47]	M. Kiewkanya, N. Jindasawat, and P. Muenchaisri, "A methodology for constructing maintainability model of object-oriented design," in *Fourth International Conference on Quality Software.* IEEE, 2004, pp. 206–213

Table A5 continued

ID	Author	Ref.	Title
S5	G.A.D. Lucca et al.	[48]	A.R. Di Lucca, Giuseppe A and Fasolino, P. Tramontana, and C.A. Visaggio, "Towards the definition of a maintainability model for web applications," in *Eighth European Conference on Software Maintenance and Reengineering*. IEEE, 2004, pp. 279–287
S6	C.V. Koten et al.	[49]	C. Van Koten and A. Gray, "An application of Bayesian network for predicting object-oriented software maintainability," *Information and Software Technology*, Vol. 48, No. 1, 2006, pp. 59–67
S7	J.H. Hayes et al.	[50]	J.H. Hayes and L. Zhao, "Maintainability prediction: A regression analysis of measures of evolving systems," in *21st International Conference on Software Maintenance (ICSM '05)*. IEEE, 2005, pp. 601–604
S8	S.C. Misra	[51]	S.C. Misra, "Modeling design/coding factors that drive maintainability of software systems," *Software Quality Journal*, Vol. 13, No. 3, 2005, pp. 297–320
S9	Y. Zhou et al.	[52]	Y. Zhou and H. Leung, "Predicting object-oriented software maintainability using multivariate adaptive regression splines," *Journal of Systems and Software*, Vol. 80, No. 8, 2007, pp. 1349–1361
S10	X. Jin et al.	[19]	X. Jin, Y. Liu, J. Ren, A. Xu, and R. Bie, "Locality preserving projection on source code metrics for improved software maintainability," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2006, pp. 877–886
S11	S.S. Dahiya et al.	[53]	S.S. Dahiya, J.K. Chhabra, and S. Kumar, "Use of genetic algorithm for software maintainability metrics' conditioning," in *15th International Conference on Advanced Computing and Communications*. IEEE, 2007, pp. 87–92
S12	M. Genero et al.	[54]	M. Genero, E. Manso, A. Visaggio, G. Canfora, and M. Piattini, "Building measure-based prediction models for UML class diagram maintainability," *Empirical Software Engineering*, Vol. 12, No. 5, 2007, pp. 517–549
S13	K. Shibata et al.	[55]	K. Shibata, K. Rinsaka, T. Dohi, and H. Okamura, "Quantifying software maintainability based on a fault-detection/correction model," in *13th Pacific Rim International Symposium on Dependable Computing*. IEEE, 2007, pp. 35–42
S14	K.K.Aggarwal et al.	[56]	K. Aggarwal, Y. Singh, A. Kaur, and R. Malhotra, "Application of artificial neural network for predicting maintainability using object-oriented metrics," *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, Vol. 2, No. 10, 2008, pp. 3552–3556
S15	Y. Thian et al.	[57]	Y. Tian, C. Chen, and C. Zhang, "AODE for source code metrics for improved software maintainability," in *Fourth International Conference on Semantics, Knowledge and Grid*. IEEE, 2008, pp. 330–335
S16	Y. Zhou et al.	[58]	Y. Zhou and B. Xu, "Predicting the maintainability of open source software using design metrics," *Wuhan University Journal of Natural Sciences*, Vol. 13, No. 1, 2008, pp. 14–20
S17	H. Yu et al.	[59]	H. Yu, G. Peng, and W. Liu, "An application of case based reasoning to predict structure maintainability," in *International Conference on Computational Intelligence and Software Engineering*. IEEE, 2009, pp. 1–5
S18	A. Sharma et al.	[60]	A. Sharma, P. Grover, and R. Kumar, "Predicting maintainability of component-based systems by using fuzzy logic," in *International Conference on Contemporary Computing*. Springer, 2009, pp. 581–591

Table A5 continued

ID	Author	Ref.	Title
S19	W. Li-jin et al.	[61]	L. Wang, X. Hu, Z. Ning, and W. Ke, "Predicting object-oriented software maintainability using projection pursuit regression," in *First International Conference on Information Science and Engineering.* IEEE, 2009, pp. 3827–3830
S20	H. Mittal et al.	[62]	H. Mittal and P. Bhatia, "Software maintainability assessment based on fuzzy logic technique," *ACM SIGSOFT Software Engineering Notes*, Vol. 34, No. 3, 2009, pp. 1–5
S21	M.O. Elish et al.	[63]	M.O. Elish and K.O. Elish, "Application of treenet in predicting object-oriented software maintainability: A comparative study," in *13th European Conference on Software Maintenance and Reengineering.* IEEE, 2009, pp. 69–78
S22	S. Rizvi et al.	[64]	S. Rizvi and R.A. Khan, "Maintainability estimation model for object-oriented software in design phase (MEMOOD)," *arXiv preprint arXiv:1004.4447*, 2010
S23	A.Kaur et al.	[65]	A. Kaur, K. Kaur, and R. Malhotra, "Soft computing approaches for prediction of software maintenance effort," *International Journal of Computer Applications*, Vol. 1, No. 16, 2010, pp. 69–75
S24	C. Jin et al.	[66]	C. Jin and J.A. Liu, "Applications of support vector mathine and unsupervised learning for predicting maintainability using object-oriented metrics," in *Second International Conference on Multimedia and Information Technology*, Vol. 1. IEEE, 2010, pp. 24–27
S25	L. CAI et al.	[67]	L. Cai, Z. Liu, J. Zhang, W. Tong, and G. Yang, "Evaluating software maintainability using fuzzy entropy theory," in *9th International Conference on Computer and Information Science.* IEEE, 2010, pp. 737–742
S26	S.O. Olatunji et al.	[68]	S.O. Olatunji, Z. Rasheed, K. Sattar, A. Al-Mana, M. Alshayeb, and E. El-Sebakhy, "Extreme learning machine as maintainability prediction model for object-oriented software systems," *Journal of Computing*, Vol. 2, No. 8, 2010, pp. 49–56
S27	P. Dhankhar et al.	[69]	P. Dhankhar, H. Mittal, and A. Mittal, "Maintainability prediction for object oriented software," *International Journal of Advances in Engineering Sciences*, Vol. 1, No. 1, 2011, pp. 8–11
S28	S.K. Dubey et al.	[70]	S.K. Dubey and A. Rana, "A fuzzy approach for evaluation of maintainability of object oriented software system," *International Journal of Computer Applications*, Vol. 49, No. 21, 2012
S29	S.K. Dubey et al.	[71]	S.K. Dubey, A. Rana, and Y. Dash, "Maintainability prediction of objec oriented software system by multilayer perceptron model," *ACM SIGSOFT Software Engineering Notes*, Vol. 37, No. 5, 2012, pp. 1–4
S30	N. Tagoug et al.	[72]	N. Tagoug, "Maintainability assessment in object-oriented system design," in *International Conference on Information Technology and e-Services.* IEEE, 2012, pp. 1–5
S31	S. Sharawat et al.	[73]	S. Sharawat, "Software maintainability prediction using neural networks," *environment*, Vol. 3, No. 5, 2012, pp. 750–755
S32	H.A. Al-Jamimi et al.	[74]	M. Al-Jamimi, Hamdi A and Ahmed, "Prediction of software maintainability using fuzzy logic," in *International Conference on Computer Science and Automation Engineering.* IEEE, 2012, pp. 702–705
S33	R. Malhotra et al.	[75]	R. Malhotra and A. Chug, "Software maintainability prediction using machine learning algorithms," *Software Engineering: An International Journal*, Vol. 2, No. 2, 2012

Table A5 continued

ID	Author	Ref.	Title
S34	T. Bakota et al.	[76]	T. Bakota, P. Hegedűs, G. Ladányi, P. Körtvélyesi, R. Ferenc, and T. Gyimóthy, "A cost model based on software maintainability," in *28th International Conference on Software Maintenance (ICSM).* IEEE, 2012, pp. 316–325
S35	Y. Dash et al.	[77]	Y. Dash, S.K. Dubey, and A. Rana, "Maintainability prediction of object oriented software system by using artificial neural network approach," *International Journal of Soft Computing and Engineering (IJSCE)*, Vol. 2, No. 2, 2012, pp. 420–423
S36	P. Hegedűs et al.	[78]	P. Hegedűs, G. Ladányi, I. Siket, and R. Ferenc, "Towards building method level maintainability models based on expert evaluations," in *Computer Applications for Software Engineering, Disaster Recovery, and Business Continuity.* Springer, 2012, pp. 146–154
S37	D. Chandra	[79]	D. Chandra, "Support vector approach by using radial kernel function for prediction of software maintenance effort on the basis of multivariate approach," *International Journal of Computer Applications*, Vol. 51, No. 4, 2012
S38	H. Aljamaan et al.	[80]	H. Aljamaan, M.O. Elish, and I. Ahmad, "An ensemble of computational intelligence models for software maintenance effort prediction," in *International Work-Conference on Artificial Neural Networks.* Springer, 2013, pp. 592–603
S39	P. Hegedűs et al.	[81]	P. Hegedűs, T. Bakota, G. Ladányi, C. Faragó, and R. Ferenc, "A drill--down approach for measuring maintainability at source code element level," *Electronic Communications of the EASST*, Vol. 60, 2013
S40	X.L. Hao et al.	[82]	X.L. Hao, X.D. Zhu, and L. Liu, "Research on software maintainability evaluation based on fuzzy integral," in *International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE).* IEEE, 2013, pp. 1279–1282
S41	F. Ye et al.	[83]	F. Ye, X. Zhu, and Y. Wang, "A new software maintainability evaluation model based on multiple classifiers combination," in *International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE).* IEEE, 2013, pp. 1588–1591
S42	M.A. Ahmed et al.	[84]	H.A. Ahmed, Moataz A and Al-Jamimi, "Machine learning approaches for predicting software maintainability: A fuzzy-based transparent model," *IET software*, Vol. 7, No. 6, 2013, pp. 317–326
S43	S.O. Olatunji et al.	[85]	S.O. Olatunji and A. Ajasin, "Sensitivity-based linear learning method and extreme learning machines compared for software maintainability prediction of object-oriented software systems," *ICTACT Journal On Soft Computing*, Vol. 3, No. 03, 2013
S44	A. Kaur et al.	[3]	A. Kaur and K. Kaur, "Statistical comparison of modelling methods for software maintainability prediction," *International Journal of Software Engineering and Knowledge Engineering*, Vol. 23, No. 6, 2013, pp. 743–774
S45	A. Mehra et al.	[86]	A. Mehra and S.K. Dubey, "Maintainability evaluation of object-oriented software system using clustering techniques," *Internationa Journal of Computers and Technology*, Vol. 5, No. 02, 2013, pp. 136–143
S46	J. Al Dallal.	[87]	J. Al Dallal, "Object-oriented class maintainability prediction using internal quality attributes," *Information and Software Technology*, Vol. 55, No. 11, 2013, pp. 2028–2048
S47	R. Malhotra et al.	[21]	R. Malhotra and A. Chug, "Application of group method of data handling model for software maintainability prediction using object oriented systems," *International Journal of System Assurance Engineering and Management*, Vol. 5, No. 2, 2014, pp. 165–173

Table A5 continued

ID	Author	Ref.	Title
S48	A. Kaur et al.	[88]	A. Kaur, K. Kaur, and K. Pathak, "A proposed new model for maintainability index of open source software," in *Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization.* IEEE, 2014, pp. 1–6
S49	A. Kaur et al.	[14]	A. Kaur, K. Kaur, and K. Pathak, "Software maintainability prediction by data mining of software code metrics," in *International Conference on Data Mining and Intelligent Computing (ICDMIC).* IEEE, 2014, pp. 1–6
S50	A. Pratap et al.	[89]	A. Pratap, R. Chaudhary, and K. Yadav, "Estimation of software maintainability using fuzzy logic technique," in *International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT).* IEEE, 2014, pp. 486–492
S51	G. Laxmi et al.	[90]	L. Geeta, A. Kavita, and B. Rizwan, "Maintainability measurement model for object oriented design," *International Journal of Advanced Research in Computer Science and Software Engineering*, Vol. 4, No. 11, 2014, pp. 945–956
S52	R. Malhotra et al.	[91]	R. Malhotra and A. Chug, "A metric suite for predicting software maintainability in data intensive applications," in *Transactions on Engineering Technologies.* Springer Netherlands, 2014, pp. 161–175
S53	S. Misra et al.	[92]	S. Misra and F. Egoeze, "Framework for maintainability measurement of web application for efficient knowledge-sharing on campus intranet," in *Computational Science and Its Applications – ICCSA 2014.* Cham: Springer International Publishing, 2014, pp. 649–662
S54	M.O. Elish et al.	[93]	M.O. Elish, H. Aljamaan, and I. Ahmad, "Three empirical studies on predicting software maintainability using ensemble methods," *Soft Computing*, Vol. 19, No. 9, 2015, pp. 2511–2524
S55	L. Kumar et al.	[94]	L. Kumar and S.K. Rath, "Neuro-genetic approach for predicting maintainability using Chidamber and Kemerer software metrics suite," in *Recent Advances in Information and Communication Technology 2015.* Cham: Springer International Publishing, 2015, pp. 31–40
S56	S.O. Olatunji et al.	[95]	S.O. Olatunji and A. Selamat, "Type-2 fuzzy logic based prediction model of object oriented software maintainability," in *Intelligent Software Methodologies, Tools and Techniques.* Cham: Springer International Publishing, 2015, pp. 329–342
S57	L. Kumar et al.	[96]	L. Kumar, D.K. Naik, and S.K. Rath, "Validating the effectiveness of object-oriented metrics for predicting maintainability," *Procedia Computer Science*, Vol. 57, 2015, pp. 798–806
S58	L. Kumar et al.	[97]	L. Kumar and S.K. Rath, "Hybrid functional link artificial neural network approach for predicting maintainability of object-oriented software," *Journal of Systems and Software*, Vol. 121, No. C, 2016, pp. 170–190
S59	A. Jain et al.	[17]	A. Jain, S. Tarwani, and A. Chug, "An empirical investigation of evolutionary algorithm for software maintainability prediction," in *Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, 2016, pp. 1–6
S60	A. Chug et al.	[98]	A. Chug and R. Malhotra, "Benchmarking framework for maintainability prediction of open source software using object oriented metrics," *International Journal of Innovative Computing, Information and Control*, Vol. 12, No. 2, 2016, pp. 615–634
S61	L. Kumar et al.	[99]	L. Kumar, K. Mukesh, and K.R. Santanu, "Maintainability prediction of web service using support vector machine with various kernel methods," *International Journal of System Assurance Engineering and Management*, Vol. 8, No. 2, 2017, pp. 205–6222

Table A5 continued

ID	Author	Ref.	Title
S62	S. Tarwani et al.	[100]	S. Tarwani and A. Chug, "Predicting maintainability of open source software using gene expression programming and bad smells," in *5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 2016, pp. 452–459
S63	S. Almugrin et al.	[32]	S. Almugrin, W. Albattah, and A. Melton, "Using indirect coupling metrics to predict package maintainability and testability," *Journal of System and Software*, Vol. 121, No. C, 2016, pp. 298–310
S64	S. Tarwani et al.	[101]	S. Tarwani and A. Chug, "Sequencing of refactoring techniques by greedy algorithm for maximizing maintainability," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2016, pp. 1397–1403
S65	L. Kumar et al.	[102]	L. Kumar, S.K. Rath, and A. Sureka, "Empirical analysis on effectiveness of source code metrics for predicting change-proneness," in *Proceedings of the 10th Innovations in Software Engineering Conference*, ISEC '17. New York, NY, USA: ACM, 2017, pp. 4–14
S66	K. Gupta et al.	[103]	G. Kanika and C. Anuradha, "Evaluation of instance-based feature subset selection algorithm for maintainability prediction," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, pp. 1482–1487
S67	S. Kundu et al.	[104]	K. Shivani and T. Kirti, "Maintainability assessment for software by using a hybrid fuzzy multi-criteria analysis approach," *Management Science Letters*, Vol. 7, 2017, pp. 255–274
S68	B.R. Reddy et al.	[16]	B.R. Reddy and O. Aparajita, "Performance of maintainability index prediction models: A feature selection based study," *Evolving Systems*, 2017
S69	L. Kumar et al.	[105]	K. Lov and K.R. Santanu, "Software maintainability prediction using hybrid neural network and fuzzy logic approach with parallel computing concept," *International Journal of System Assurance Engineering and Management*, Vol. 8, No. S2, 2017, pp. 1487–1502
S70	L. Kumar et al.	[106]	L. Kumar, A. Krishna, and S.K. Rath, "The impact of feature selection on maintainability prediction of service-oriented applications," *Service Oriented Computing and Applications*, Vol. 11, No. 2, 2017, pp. 137–161
S71	L. Kumar et al.	[107]	L. Kumar, S.K. Rath, and A. Sureka, "Using source code metrics and multivariate adaptive regression splines to predict maintainability of service oriented software," in *18th International Symposium on High Assurance Systems Engineering (HASE)*, 2017, pp. 88–95
S72	L. Kumar et al.	[108]	L. Kumar, S.K. Rath, and A. Sureka, "Using source code metrics to predict change-prone web services: A case-study on ebay services," in *IEEE Workshop on Machine Learning Techniques for Software Quality Evaluation – MaLTeSQuE*. IEEE, 2017, pp. 1–7
S73	R. Malhotra et al.	[109]	R. Malhotra and R. Jangra, "Prediction and assessment of change prone classes using statistical and machine learning techniques," *Journal of Information Processing Systems*, Vol. 13, No. 4, 2017, pp. 778–804
S74	G. Szoke et al.	[110]	G. Szőke, G. Antal, C. Nagy, R. Ferenc, and T. Gyimóthy, "Empirical study on refactoring large-scale industrial systems and its effects on maintainability," *Journal of Systems and Software*, Vol. 129, 2017, pp. 107–126
S75	Y. Gokul et al.	[111]	Y. Gokul and M. Gopal, "An authoritative method using fuzzy logic to evaluate maintainability index and utilizability of software," *Advances in Modelling and Analysis B*, Vol. 60, No. 3, 2017, pp. 566–580

Table A5 continued

ID	Author	Ref.	Title
S76	P. Hegedűs et al.	[112]	P. Hegedűs, I. Kádár, R. Ferenc, and T. Gyimóthy, "Empirical evaluation of software maintainability based on a manually validated refactoring dataset," *Information and Software Technology*, Vol. 95, No. 1, 2018, pp. 313–327
S77	L. Kumar et al.	[113]	L. Kumar and S. Ashish, "A comparative study of different source code metrics and machine learning algorithms for predicting change proneness of object oriented systems," *arXiv preprint arXiv:1712.07944*, 2018
S78	G. Scanniello et al.	[114]	G. Scanniello, C. Gravino, M. Genero, J.A. Cruz-Lemus, and G. Tortora, "On the impact of UML analysis models on source-code comprehensibility and modifiability," *ACM Trans. Softw. Eng. Methodol.*, Vol. 23, No. 2, 2014, pp. 13:1–13:26
S79	A.M. Fernández-Sáez et al.	[115]	A.M. Fernández-Sáez, M.R.V. Chaudron, M. Genero, and I. Ramos, "Are forward designed or reverse-engineered UML diagrams more helpful for code maintenance?: A controlled experiment," in *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering.* New York, NY, USA: ACM, 2013, pp. 60–71
S80	G. Scanniello et al.	[116]	G. Scanniello, C. Gravino, G. Tortora, M. Genero, M. Risi, J.A. Cruz-Lemus, and G. Dodero, "Studying the effect of UML-based models on source-code comprehensibility: Results from a long-term investigation," in *Proceedings of the 16th International Conference on Product-Focused Software Process Improvement*, Vol. 9459, New York, 2015, pp. 311–327
S81	A.M. Fernández-Sáez et al.	[117]	A.M. Fernández-Sáez, M. Genero, D. Caivano, and M.R. Chaudron, "Does the level of detail of UML diagrams affect the maintainability of source code?: A family of experiments," *Empirical Software Engineering*, Vol. 21, No. 1, 2016, pp. 212–259
S82	G. Scanniello et al.	[118]	G. Scanniello, C. Gravino, M. Genero, J.A. Cruz-Lemus, G. Tortora, M. Risi, and G. Dodero, "Do software models based on the UML aid in source-code comprehensibility? Aggregating evidence from 12 controlled experiments," *Empirical Software Engineering*, Vol. 23, No. 5, 2018, pp. 2695–2733

Table A6. Results of data extraction from nine databases

ID	Ref.	MQ1	MQ2		MQ3	MQ4	MQ5	MQ6	MQ8
		Publication year	Publication source	Publication channel	Research type	Empirical approach	Software application type	Dataset	Technique
S1	[44]	2000	CRE	Conference	SP	Ex	POA	ANSI C programs	PR
S2	[45]	2003	SEKE	Conference	ER	HbE	OOA	QUES, UIMS	GRNN, WNN
S3	[46]	2003	METRICS	Symposium	SP	Ex	OOA	Different OO application domains	MLR
S4	[47]	2004	QSIC	Conference	SP	Ex	OOA	Different OO application domains	DA
S5	[48]	2004	CSMR	Conference	SP	CS	WbA	Different WA Freeware applications	WF
S6	[49]	2005	IST	Journal	ER	HbE	OOA	QUES, UIMS	BN, RT, BE, SS
S7	[50]	2005	ICSM	Conference	SP	Ex	OOA	Different software student projects	MLR
S8	[51]	2005	SQJ	Journal	SP	Ex	OOA	C++ open source software	MLR
S9	[52]	2006	JSS	Journal	ER	HbE	OOA	QUES, UIMS	MARS, MLR, ANN, RT, SVR
S10	[19]	2006	AJCAI	Conference	SP	HbE	POA	Medical Imaging System	GMM, SVM-RBF, DT
S11	[53]	2007	ICACC	Conference	SP	Ex	NI	Different software data	FL
S12	[54]	2007	ESE	Journal	SP	Ex	OOA	Different OO application domains	MLR
S13	[55]	2007	ISPRDC	Symposium	SP	CS	NI	Different software projects	SM
S14	[56]	2008	IJCEACIE	Journal	ER	HbE	OOA	QUES, UIMS	ANN
S15	[57]	2008	ICSKG	Conference	SP	HbE	POA	Medical imaging System	AODE, SVM-LIN, NB, BN, RF, KNN, C4.5, OneR, RBF
S16	[58]	2008	WUJNS	Journal	SP	Ex	OOA	Java open source software	MLR
S17	[59]	2009	CISE	Conference	SP	CS	NI	Data from software design	CBR
S18	[60]	2009	ICCC	Conference	SP	CS	CbA	Billing system	FL
S19	[61]	2009	ICISE	Conference	ER	HbE	OOA	QUES, UIMS	PPR, ANN, MARS

Table A6 continued

ID	Ref.	MQ1	MQ2		MQ3	MQ4	MQ5	MQ6	MQ8
		Publication year	Publication source	Publication channel	Research type	Empirical approach	Software application type	Dataset	Technique
S20	[62]	2009	SIGSOFT	Conference	SP	HbE	POA	Aggarwal	FL
S21	[63]	2009	CSMR	Conference	ER	HbE	OOA	QUES, UIMS	TreeNet
S22	[64]	2010	JC	Journal	SP	HbE	OOA	BIS	MLR
S23	[65]	2010	IJCA	Journal	ER	HbE	OOA	QUES, UIMS	FFNN, FIS, ANFIS, GRNN, RBF
S24	[66]	2010	ICMIT	Conference	ER	Ex	OOA	Software developed by students	SVM
S25	[67]	2010	ICCIS	Conference	SP	CS	NI	Different software products	FET
S26	[68]	2010	JC	Journal	SP	HbE	OOA	QUES, UIMS	ELM
S27	[69]	2011	IJAES	Journal	SP	CS	OOA	ATM System	FL
S28	[70]	2012	IJCA	Journal	SP	Ex	OOA	OO software systems	FL
S29	[71]	2012	SIGSOFT	Conference	ER	HbE	OOA	QUES, UIMS	MLP
S30	[72]	2012	ICITeS	Conference	SP	CS	OOA	Real BIS	LR
S31	[73]	2012	IJERA	Journal	ER	HbE	OOA	UIMS	ANN
S32	[74]	2012	ICSESS	Conference	ER	HbE	OOA	QUES, UIMS	MFL
S33	[75]	2012	SEIJ	Journal	ER	HbE	OOA	QUES, UIMS	GMDH, GA, PNN
S34	[76]	2012	ICSM	Conference	SP	CS	OOA	Jedit, Log4j Java projects	PD
S35	[77]	2012	IJSCE	Journal	ER	HbE	OOA	UIMS	MLP
S36	[78]	2012	ASEA-DRBC	Conference	SP	Ex	OOA	Jedit, Industrial software product	LR, ANN, DT
S37	[79]	2012	IJCA	Journal	ER	HbE	OOA	QUES, UIMS	SVM-RBF
S38	[80]	2013	IWCANN	Conference	SP	HbE	OOA	QUES, UIMS	MLP, RBF, SVM, M5P, Ensemble
S39	[81]	2013	IWSQM	Workshop	SP	HbE	OOA	Jedit	PD
S40	[82]	2013	QR2MSE	Conference	SP	Ex	NI	Virtual maintenance system	FIT
S41	[83]	2013	QR2MSE	Conference	SP	Ex	OOA	C++ open source system	DT, BPNN, SVM, Bagging
S42	[84]	2013	ESSE	Journal	SP	HbE	OOA	QUES, UIMS	MFL, ANFIS, PNN, RBF, SVM

Table A6 continued

ID	Ref.	MQ1	MQ2		MQ3	MQ4	MQ5	MQ6	MQ8
		Publication year	Publication source	Publication channel	Research type	Empirical approach	Software application type	Dataset	Technique
S43	[85]	2013	ICTACT	Journal	ER	HbE	OOA	QUES, UIMS	SBLLM, ELM
S44	[3]	2013	IJSEKE	Journal	ER	HbE	OOA	QUES, UIMS	MLR, LMSR, PaceR, PPR, IR, RegByDisc, GPR, MLP, RBF, AR, GRNN, GMDH, SVR, FSC, DS, ANFIS, K*, M5P, LWL, Bagging, KNN, REPTree, RF, ES, CR, DTable, M5R
S45	[86]	2013	IJCT	Journal	ER	HbE	OOA	QUES	KMC, XMC
S46	[87]	2013	IST	Journal	SP	Ex	OOA	Different OO application domains	LgR
S47	[21]	2014	IJSAE	Journal	ER	HbE	OOA	FLM, EASY	GMDH, FF3LBPN, GRNN
S48	[88]	2014	ICRITO	Conference	SP	HbE	OOA	Lucence	SS, BE
S49	[14]	2014	ICDMIC	Conference	ER	HbE	OOA	Lucence, JHotdraw, JEdit, JTreeview	NB, BN, LgR, MLP, RF
S50	[89]	2014	ICICT	Conference	SP	Ex	OOA	Private software system	FL
S51	[90]	2014	IJARCSSE	Journal	SP	HbE	OOA	BIS	ML
S52	[91]	2014	–	Chapter	SP	HbE	OOA	FLM, EASY, ABP, SMS, IMS	MLR, BPNN, KN, FFNN, GRNN
S53	[92]	2014	ICCSA	Conference	SP	Ex	WbA	Different WA domains	LR
S54	[93]	2015	JSC	Journal	ER	HbE	OOA	QUES, UIMS, Vssplugin, PeerSim	MLP, RBF, SVM, M5P, DT, LgR, KMC, GEP, MV, NL, Boosting, Bagging, AVG, WT, BTE
S55	[94]	2015	IC2IT	Conference	SP	HbE	OOA	QUES, UIMS	ANN, Neuro-GA
S56	[95]	2015	ICISMTT	Conference	SP	HbE	OOA	UIMS	T2FLS
S57	[96]	2015	PCS	Journal	ER	HbE	OOA	QUES, UIMS	Neuro-GA
S58	[97]	2016	JSS	Journal	SP	HbE	OOA	QUES, UIMS	FGA, AFGA, FPSO, MFPSO, FCSA

Table A6 continued

ID	Ref.	MQ1	MQ2		MQ3	MQ4	MQ5	MQ6	MQ8
		Publication year	Publication source	Publication channel	Research type	Empirical approach	Software application type	Dataset	Technique
S59	[17]	2016	SCEECS	Conference	ER	HbE	OOA	jTDS, jWebUnit , jXLS, SoundHelix	GA, DTable, RBF, BN, SMO
S60	[98]	2016	IJICIC	Journal	ER	HbE	OOA	Drumkit, OpenCV, Abdera, Ivy, Log4j, JEdit , JUnit	LR, M5R, DT, SVM, K*, Bagging, JERN, BPNN, KN, PNN, GMDH, GRNN, GGAL
S61	[99]	2016	IJSAEM	Journal	ER	HbE	SOA	5 versions of eBay web service system	SVM-LIN, SVM-SIG, SVM-RBF
S62	[100]	2016	ICRITO	Conference	ER	HbE	OOA	jTDS, Jchess, ArtOfIllusion, OrDrumbox	GEP, DFT, SVM, LR, MLP, RBF
S63	[32]	2016	JSS	Journal	ER	HbE	OOA	Camel, JEdit, Tomcat, JHotDraw	MLR
S64	[101]	2016	ICACCI	Conference	SP	HbE	OOA	jtds	GdA
S65	[102]	2017	HASE	Symposium	ER	HbE	OOA	Eclipse software application	LR, NB, ELM-LIN, ELM-PLY, BTE, MV, SVM-SIG, ELM-RBF, SVM-LIN, SVM-RBF
S66	[103]	2017	ICACCI	Conference	ER	Ex	OOA	Apache Jackrabbit, Light Weight Java Game Library	LR, Cubist, Lasso, Elastic Net, RF
S67	[104]	2017	MSL	Journal	SP	Ex	NI	3 software products	FL

Table A6 continued

ID	Ref.	MQ1	MQ2		MQ3	MQ4	MQ5	MQ6	MQ8
		Publication year	Publication source	Publication channel	Research type	Empirical approach	Software application type	Dataset	Technique
S68	[16]	2017	ES	Journal	ER	HbE	OOA	Art of illusion, Camel, Eclipse, Free mind, Games, Gantt, Geoxygene, Ivy, Jabref, Jajuk, Jasper reports, Javaml, Jfree ant, Jfree chart, Jgap, Jmt, Jnetpcap, Lucene, Mallet, Pandora, POI, Sglj, Tree view, Ujac, Workzen, Xalan	MLR, MLP, SVR, M5P
S69	[105]	2017	IJSAEM	Journal	SP	HbE	OOA	UIMS, QUES	Neuro Fuzzy
S70	[106]	2017	SOCA	Journal	ER	HbE	SOA	5 versions of eBay web service	SVM-LIN, SVM-SIG, SVM-RBF
S71	[107]	2017	HASE	Symposium	ER	HbE	SOA	5 versions of eBay web services	MARS, MLR, SVM
S72	[108]	2017	MaLTeSQuE	Workshop	SP	HbE	SOA	5 versions of eBay web services	LSSVM-LIN, LSSVM-RBF, LSSVM-SIG
S73	[109]	2017	JIPS	Journal	ER	HbE	OOA	Art-of-Illusion, Sweet-Home-3D	LgR, RF, Bagging, AdaBoost, MLP, BN, NB, LogitBoost, J48, NNge
S74	[110]	2017	JSS	Journal	ER	Ex	OOA	Industrial systems	PD
S75	[111]	2017	AMSE/IIETA	Journal	SP	Ex	OOA	Private data sources	MFL
S76	[112]	2018	IST	Journal	SP	Ex	OOA	antlr4, junit, mapdb, mcMMO, mct, oryx, titan	Statistical

Table A6 continued

		MQ1	MQ2		MQ3	MQ4	MQ5	MQ6	MQ8
ID	Ref.	Publication year	Publication source	Publication channel	Research type	Empirical approach	Software application type	Dataset	Technique
S77	[113]	2018	arXiv	Journal	ER	HbE	OOA	Compare, webdav, debug, update, core, swt, team, pde, ui, jdt	LR, PR, LgR, DT, SVM-LIN, SVM-PLY, SVM-RBF, ELM-LIN, ELM-PLY, ELM-RBF, LSSVM-LIN, LSSVM-PLY, LSSVM-RBF, NGD, NGDM, BTE, NGDA, NNM, MVE, NDTF, NLM
S78	[114]	2014	TOSEM	Journal	SP	Ex	OOA	2 private systems (the selling of CDs/DVDs in a music shop and the booking of theater tickets)	Statistical
S79	[115]	2015	IST	Journal	SP	Ex	OOA	OO application domain (Sports center application which was created as part of the Master's Thesis of a student from the University of Castilla-La Mancha)	Statistical
S80	[116]	2015	PROFES	Conference	SP	Ex	OOA	A chunk of a system music shop software and a chunk of a theater ticket reservation system implemented in Java, JHotDraw	Statistical

Table A6 continued

ID	Ref.	MQ1	MQ2		MQ3	MQ4	MQ5	MQ6	MQ8
		Publication year	Publication source	Publication channel	Research type	Empirical approach	Software application type	Dataset	Technique
S81	[117]	2016	ESE	Journal	SP	Ex	OOA	2 systems (a library application from which a user can borrow books and a sport center application from which users can rent services)	Statistical
S82	[118]	2018	ESE	Journal	SP	Ex	OOA	OO application domains (music shop and theater ticket reservation applications)	Statistical

Acronyms used in Table:
• **Research type acronyms**: Solution Proposal (SP), Evaluation Research (ER)
• **Empirical approach acronyms**: History-based Evaluation (HbE), Experiment (Ex), Case study (CS)
• **Software application type acronyms**: Object Oriented Applications (OOA), Procedural Oriented Applications (POA), Web-based Applications (WbA), Component-based Application (CbA), Service Oriented Applications (SOA), Not Identified (NI).
• **Publication source acronyms**: Conference on Reverse Engineering (CRE), International Conference on Software Engineering & Knowledge Engineering (SEKE), International Software Metrics Symposium (METRICS), International Conference on Quality Software (QSIC), European Conference on Software Maintenance and Reengineering (CSMR), Information and Software Technology (IST), International Conference on Software Maintenance (ICSM), Software Quality Journal (SQJ), The Journal of Systems & Software (JSS), International Conference on Advanced Computing and Communications (ICACC), Empirical Software Engineering (ESE), IEEE International Symposium on Pacific Rim Dependable Computing (ISPRDC), International Journal of Computer, Electrical, Automation, Control and Information Engineering (IJCEACIE),International Conference on Semantics, Knowledge and Grid (ICSKG), Wuhan University Journal of Natural Sciences (WUJNS), International Conference on Computational Intelligence and Software Engineering (CISE), Computing International Conference on Information Science and Engineering (ICISE), SIGSOFT Software Engineering Notes (SIGSOFT), Journal of Computing (JC), International Journal of Computer Applications (IJCA), International Conference on Multi-Media and Information Technology (ICMIT), International Conference on Computer and Information Science (ICCIS), International Journal of Advances in Engineering Sciences (IJAES), International Conference on Information Technology and e-Services (ICITeS), International Journal of Engineering Research and Applications (IJERA), International Conference on Software Engineering and Service Science (ICSESS), Software engineering: an international Journal (SEIJ), International Journal of Soft Computing and Engineering (IJSCE), International Workshop on Software Quality and Maintainability (IWSQM), International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE), Special Issue on Empirical Studies in Software Engineering (ESSE), ICTACT Journal on Soft Computing (ICTACT), International Journal of Software Engineering and Knowledge Engineering (IJSEKE),

International Journal of Computers & Technology (IJCT), International Journal of System Assurance Engineering and Management (IJSAEM), International Conference on Reliability, Infocom Technologies and Optimization (ICRITO), International Conference on Data Mining and Intelligent Computing (ICDMIC), International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), International Conference on Computational Science and Its Applications (ICCSA), The Journal of Soft Computing (JSC), International Conference on Intelligent Software Methodologies, Tools, and Techniques (ICISMTT), Procedia Computer Science (PCS), International Work-Conference on Artificial Neural Networks (IWCANN), Computer Applications for Software Engineering, Disaster Recovery, and Business Continuity (ASEA-DRBC), Australasian Joint Conference on Artificial Intelligence (AJCAI), International Conference on Contemporary Computing (ICCC), International Conference on Computing and Information Technology (IC2IT), IEEE Students' Conference on Electrical, Electronics and Computer Science (SCEECS), International Journal of Innovative Computing, Information and Control (IJICIC), International Conference on Reliability Infocom Technologies and Optimization (ICRITO), International Conference on Advances in Computing, Communications and Informatics (ICACCI), International Symposium on High Assurance Systems Engineering (HASE), Management Science Letters (MSL), Evolving System: An Interdisciplinary Journal for Advanced Science and Technology (ES), Service Oriented Computing and Applications (SOCA), Workshop on Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE), Journal of Information Processing Systems (JIPS), arXiv Repository of electronic preprints (arXiv), ACM Transactions on Software Engineering and Methodology (TOSEM), International Conference on Product-Focused Software Process Improvement (PROFES), Association for the Advancement of Modelling and Simulation Techniques in Enterprises/International Information and Engineering Technology Association (AMSE/IIETA).

- **Technique acronyms**: Linear Regression (LR), Multiple Linear Regression (MLR), Logistic Regression (LgR), Backward Elimination (BE), Stepwise Selection (SS), Multiple Adaptive Regression Splines (MARS), Projection Pursuit Regression (PPR), Polynomial Regression (PR), Least Median of Squares Regression (LMSR), Pace Regression (PaceR), Isotonic Regression (IR), Regression By Discretization (RegByDisc), Additive Regression (AR), Gaussian Process Regression (GPR), Least Absolute Shrinkage and Selection Operator (Lasso), Probability Density function (PD), Gaussian Mixture Model (GMM), Discriminant Analysis (DA), Weighted Functions (WF), Stochastic Model (SM), Artificial Neural network (ANN), Multilayer Perceptron (MLP), Radial Basis Function Network (RBF), Probabilistic Neural Network (PNN), Group Method of Data Handling (GMDH), General Regression Neural Network (GRNN), Feed Forward Neural Network (FFNN), Back Propagation Neural Network (BPNN), Kohonen Network (KN), Ward Neural Network (WNN), Feed Forward 3-Layer Back Propagation Network (FF3LBPN), Extreme Learning Machines (ELM), Sensitivity Based Linear Learning Method (SBLLM), Neuro-Genetic algorithm (Neuro-GA), Functional Link Artificial Neural Network (FLAAN) with Genetic Algorithm (FGA), Adaptive FLANN-Genetic (AFGA), FLANN-Particle Swarm Optimization (FPSO), Modified-FLANN Particle Swarm Optimization (MFPSO), FLANN-Clonal Selection Algorithm (FCSA), ELM with linear (ELM-LIN), ELM with polynomial (ELM-PLY), ELM with Radial Basis Function kernels (ELM-RBF), ANN with Levenberg Marquardt method (NLM), GRNN with Genetic Adaptive Learning (GGAL), Jordan Elman Recurrent Network (JERN), ANN with normally Gradient descent method (NGD), ANN with Gradient descent with momentum (NGDM), ANNf with Gradient descent with adaptive learning rate (NGDA) method, ANN with Quasi-Newton method (NNM), Kstar (K*), Locally Weighted Learning (LWL), *k*-Nearest Neighbor (IBK or KNN), Nearest-Neighbor-like algorithm that uses non-nested generalized exemplars (NNge), Bayesian Networks (BN), Decision Tree (DT), Regression tree (RT), M5 for inducing trees of regression models (M5P), Random Forest (RF), Decision Stump (DS), Reduced Error Pruned Tree (REPTree), Decision Tree Forest (DFT), Genetic Expression programming (GEP), Case-Based Reasoning (CBR), Genetic Algorithm (GA), Greedy algorithm (GdA), Naive-Bayes (NB), Aggregating One-Dependence Estimators (AODE), Support Vector Machine (SVM), Support Vector Regression (SVR), Sequential Minimal Optimization (SMO), SVM with radial basis function kernel (SVM-RBF), SVM with linear kernel (SVM-LIN), SVM with sigmoid kernel (SVM-SIG), Least Square Support Vector Machine (LSSVM) with linear kernel (LSSVM-LIN), LSSVM with radial basis function kernel (LSSVM-RBF), SVM with Polynomial Kernel (SVM-PLY), LSSVM with sigmoid kernel (LSSVM-SIG), LSSVM with Polynomial Kernel (LSSVM-PLY), Fuzzy Logic (FL), Adaptive Neuro-Fuzzy Inference Systems (ANFIS), Fuzzy Inference Systems (FIS), Type-2 fuzzy logic system (T2FLS), Mamdani-Based Fuzzy Logic (MFL), Fuzzy Entropy Theory (FET), Fuzzy Subtractive Clustering (FSC), Fuzzy Integral theory (FIT), Decision Table (DTable), Conjunctive Rule Learner (CR), M5 Rules (M5R), K-Means Clustering algorithm (KMC), X-Means Clustering algorithm (XMC), Ensemble Selection (ES), Average-based ensemble (AVG), Weighted-based ensemble (WT), Best-in-training-based ensemble (BTE), Majority-voting ensemble (MV), Non-linear ensemble (NL), Nonlinear Ensemble Decision Tree Forest (NDTF), Adaptive Boosting (AdaBoost).

• **Dataset acronyms**: UIMS (User Interface Management System), QUES (Quality Evaluation System), VSSPLUGIN (Visual Source Safe PLUGIN), PeerSim (Peer-to-Peer Simulator), MIS (Medical imaging system), FLM (File Letter Monitoring System), EASY (EASY Classes Online Services collection), SMS (Student Management System), IMS (Inventory Management System) and APB (Angel bill printing), Bank Information System (BIS).

Table A7. Distribution of predictors used as independent variables

Predictors	Supported studies	# of studies (percent)
Chidamber and Kemerer (C&K) measures, such as: coupling between object (CBO), depth of inheritance tree (DIT), number of children (NOC), weighted methods per Class (WMC), response for a class (RFC), lack of cohesion in methods (LCOM)	S2, S6, S8, S9, S14, S16, S19, S21, S23, S24, S26, S28, S29, S31, S32, S34, S33, S35, S37, S38, S41, S42, S43, S44, S45, S46, S47, S48, S49, S52, S54, S55, S56, S57, S58, S59, S60, S61, S62, S64, S65, S66, S68, S69, S70, S71, S72, S73, S75, S77	50 (61%)
Li and Henry (L&H) measures, such as: message passing coupling (MPC), data abstraction coupling (DAC), number of local methods (NOM), SIZE1 (LOC calculated by counting the number of semicolons in a class), SIZE2 (Number of properties including the attributes and methods defined in a class)	S2, S6, S9, S14, S19, S21, S23, S24, S26, S29, S31, S32, S33, S35, S37, S38, S42, S43, S44, S45, S46, S47, S48, S49, S54, S56, S57, S58, S61, S65, S69, S70, S71	33 (40%)
Class diagram measures, such as measures related to: – method level such as: number of methods, average number of methods per class, number of foreign methods accessed, number of local methods accessed, number of constructors, etc. – attribute level such as: number of attributes, average number of attributes per class, number of attributes added, etc. – class level such as: number of classes, classes changed, classes added, etc. – relationships such as: number of generalisations, number of associations, number of aggregations, number of dependencies.	S3, S4, S7, S8, S12, S16, S22, S27, S36, S39, S41, S46, S48, S49, S51, S59, S62, S65, S66, S68, S72, S73, S74, S77	24 (29%)
Source code size measures, different lines of code (LOC) measures, such as: source lines of code, logical lines of code, total lines of code, etc.	S7, S8, S10, S15, S20, S34, S36, S39, S41, S46, S48, S49, S62, S65, S66, S68, S73, S74, S76, S77	20 (24%)
McCabe complexity measure (McCabe)	S1, S20, S27, S34, S36, S39, S48, S49, S50, S52, S61, S70, S71, S72, S73, S74, S76	17 (21%)
Software quality attributes, such as: understandability, documentation quality, readability of source code, testability, etc.	S11, S17, S18, S25, S40, S50, S63, S67	8 (10%)
Martin's measures, such as: abstractness (A), the distance from the main sequence (D), and the normalized distance from the main sequence (Dn), etc.	S48, S49, S59, S60, S63, S66	6 (7%)
Halstead measures, such as: number of distinct operators, number of distinct operands, total number of occurrences of operators, total number of occurrences of operands, length (N) of a program, program volume (V), etc.	S8, S10, S15, S48, S49	5 (6%)
Brito e Abreu and Carapuça (BA&C) measures, such as: method hiding factor, polymorphism factor, etc.	S8, S16, S62, S64, S68	5 (6%)

Table A7 continued

Predictors	Supported studies	# of studies (percent)
Factors such as: origin of UML diagrams, level of detail of UML diagrams, method (analysis models and source code, and source code alone), models (software models plus source code without comment), ability, and source code without comments.	S78, S79, S80, S81, S82	5 (6%)
Coding rule measures, such as: number of serious coding rule violations, number of suspicious coding rule violations, number of coding style issues, etc.	S34, S36, S74	3 (4%)
QMOOD measures, such as: data access metric (DAM), measure of aggression (MOA), method of functional abstraction (MFA), etc.	S59, S60, S66	3 (4%)
Maintainability index (MI) measure	S7, S52	2 (2%)
Web-based application (WbA) measures, such as: total web page, server script, web page control structure, client page, web control coupling, server page, etc. class diagram measures	S5, S53	2(2%)
Jensen measures	S10, S15	2 (2%)
Effort measures, such as: coding effort, design effort, requirement effort, effort integration, ratio of requirement effort and design effort to coding effort (RDCRatio), etc.	S7, S30	2 (2%)
Module level measures, such as: percentage of modules changed, module level information flow, etc.	S1, S7	2 (2%)
Sequence diagram measures, such as measures related to: scenarios (number of scenarios), – messages (average number of return messages, weighted messages between objects, average number of the directly dispatched messages, etc. – conditions (average number of condition messages)	S4	1 (1%)
Lorenz and Kidd (L&K) measures, such as: average method size and coupling factor, etc.	S8	1 (1%)
Fault measures, such as: number of detected faults and number of corrected faults, etc.	S13	1 (1%)
Database measures, such as: number of data base connections and the schema complexity to comment ratio, etc.	S52	1 (1%)

Table A8. Acronyms of successful predictors

Acronym	Description	Acronym	Description
ACLOC	Average class lines of code	NAssoc	Number of associations
AIF	Attribute inheritance factor	NC	Number of classes
AMLOC	Average method lines of code	NClienP	Number of client pages
AVPATHS	Average depth of paths	NHD	Normalized Hamming distance
B	Number of bugs (Halstead)	NEWLCOM3	New lack of cohesion in methods 3
Ca	Afferent coupling	NPM	Number of Public methods
CAMC	Cohesion among methods in a class	NDep	Number of dependencies

Table A8 continued

Acronym	Description	Acronym	Description
CBO_IUB	Coupling between object (CBO) – Is used by attributes or methods of class	NGen	Number of generalisation
CBO_U	CBO – Using by the methods of class	NGenH	Number of generalisation hierarchies
CC	Cyclomatic complexity	NM	Number of methods
CDENS	Control DENSity	NOC	Number of children
Ce	Efferent coupling	NODBC	Number of data base connections
CLOC	Comments lines of code	NOM	Number of local methods
ClS	Client scripts	NPAVGC	Average number of parameters per method
COF	Coupling factor	NServP	Number of server pages
Coh	Cohesion	NWebP	Number of web pages
Command	Number of commands	OCmax	Maximum operation complexity
CONS	Number of constructors	OCMEC	Other class method export coupling
CSA	Average number of attributes per class	OL2	The average strength of the attribute
CSO	Average number of methods per class	OSAVG	Average complexity per method
Cyclic	Number of cyclic dependencies	OSmax	Maximum operation size
DAC	Data abstraction coupling	PCCC	Path connectivity class cohesion
DAM	Data access metric	POF	Polymorphism factor
DCd	Degree of cohesion-direct	PPPC	Percentage public/protected members
DCi	Degree of cohesion-indirect	Query	Number of query
Dcy	Number of dependencies	RFC	Response for a class
Dcy*	Number of transitive dependencies	RDCRatio	Ratio of Requirement Effort and Design Effort to Coding Effort
DIT	Depth of inheritance tree	SCCR	Schema complexity to comment Ratio
Inner*	Number of inner classes	SCOM	Class cohesion metric
LCC	Loose class cohesion	SIZE1/LOC	Line of code
LCOM	Lack of cohesion in methods	SIZE2	Number of Properties
LLOC	Logical lines of code	SLoc	Source lines of code
LSCC	Low-level design similarity-based class cohesion	SS	Server scripts
MaxDIT	Maximum depth of inheritance tree	STAT	Number of STATements
MI	Maintainability index	SWMC	Average weighted methods per class
MIF	Method inheritance factor	TCC	Total cyclomatic complexity
MOA	Measure of aggregation	TCC	Tight class cohesion
MPC	Message passing coupling	TL	Total languages
n	Vocabulary size (Halstead)	TLOC	Total LOC
N	Program length (Halstead)	TWP	Total web page
NA/NOA	Number of attributes	TWPDC	Total web page data coupling
NAA	Number of attributes added	TWPR	Total web page relationships
NAgg	Number of aggregations	V	Program volume (Halstead)
NAggH	Number of aggregations hierarchies	WMC	Weighted methods per Class
NAggR	Number of aggregation relationships	WO	WebObject

Table A9. Prediction techniques and accuracy criteria per dependent variable topics

Topic	ID	Prediction technique per category	Accuracy criteria
Change	S2	ANN	R-squared, R, MSE, MAE, MinAE, MaxAE
	S6	BN, DT, RA	MaxMRE, MMRE, Pred(25), Pred(30), sum Ab. Res, med. Ab. Res, Std. Ab. Res
	S9	RA, ANN, DR, SVM/R	MaxMRE, MMRE, Pred(25), Pred(30), sum ARE, Med. ARE, Std. ARE
	S10	GMM, SVM/R, DT	WAP, recall
	S14	ANN	MARE, R, MRE
	S15	ANN, SVM/R, DT, BN, CBR	WAP, WARec
	S19	ANN, RA	RMSE
	S23	ANN, FNF	MARE, MRE, R
	S24	SVM/R	MARE, MRE, R
	S26	ANN	MaxMRE, MMRE, Pred(25), Pred(30), Sum Ab. Res, Med. Ab. Res, Std. Ab. Res
	S29	ANN	R-squared, R, MAE, minAE, maxAE
	S30	RA	–
	S31	ANN	–
	S32	FNF	RMSE, NRMSE, MMRE
	S33	ANN, EA	MaxMRE, MMRE, MARE, Pred(25), Pred(30), Pred(75)
	S35	ANN	R, MAE
	S37	SVM/R	MARE, MRE, R
	S38	ANN, SVM/R, DT, EM	MMRE, Std. MRE, Pred(30)
	S42	FNF, ANN, SVM/R	NRMSE, MMRE, Pred(25), Pred(30).
	S43	ANN	MaxMRE, MMRE, Pred(25), Pred(30), Sum Ab. Res., Med. Ab. Res., Std. Ab. Res.
	S44	RA, ANN,SVM/R, DT, FNF, CBR, IRB	MaxMRE, MMRE, Pred(25), Pred(30), Sum ARE, Med. ARE, Std. ARE, RMSE
	S45	CM	Qout, Nit, cut-off
	S47	ANN	MMRE, Pred(25), Pred(30), % under, % over
	S48	RA	R-squared, R
	S49	RA, ANN, BN	Recall, Precision, ROC area
	S52	RA, ANN	MaxMRE, MMRE, Pred(25)
	S54	RA, ANN, SVM/R, DT, EA, CM	MMRE, Std. MRE, Pred(30), CCR, AUC
	S55	ANN	MMRE, MARE, MAE, RMSE, SEM
	S57	ANN	MAE, MARE, RMSE, SEM, MMRE, e, é
	S58	ANN	MAE, R, MMRE, e, é
	S59	ANN, SVM/R, BN, EA, IRB	MAE, RMSE

Table A9 continued

Topic	ID	Prediction technique per category	Accuracy criteria
Change	S60	RA, ANN, SVM/R, DT, FNF, CBR, IRB	MAE, RMSE, Pred(25), Pred(75)
	S61	SVM/R	Precision, Recall, F-measure, Specificity, Accuracy, AUC
	S62	RA, ANN, SVM/R, DT, EA	MAE, RMSE
	S64	EA	–
	S65	RA, ANN, SVM/R, EM, BN	Accuracy, AUC
	S66	RA	MAE, RMSE, Accuracy
	S69	FNF	Performance index
	S70	SVM/R	F-measure, Accuracy
	S71	RA, SVM/R	Accuracy, Recall, Precision
	S72	SVM/R	F-measure, Accuracy
	S73	RA, ANN, DT, BN, CBR	Sensitivity, Specificity, ROC, cutoff
	S74	PD	–
	S77	RA, ANN, SVM/R, DT	Accuracy, F-measure
Expert opinion	S11, S18, S20, S25, S27, S28, S50, S67	FNF	–
	S36	RA, ANN, DT	MAE
	S41	ANN, DT, SVM/R	TPR, FPR, Precision, Recall, F1 score, AUC
Maintainability index	S8	RA	R-squared, R, adjusted R-squared, Std. EE
	S16	RA	R-squared, MARE, MMRE
	S39	PF	R
	S40	FNF	–
	S48	RA	R-squared, R
	S68	RA, ANN, DT, SVM/R	AOC, StdMRE, MMRE, Pred(30)
	S75	FNF	–
	S76	Statistical	Rs
Maintainability level	S4	DA	Accuracy
	S22	RA	R-squared, R, adjusted R-squared, Std.EE,
	S51, S53	RA	Rs
	S78, S80, S82	Statistical	–
Maintainability time	S3	RA	MMRE, qMRE, Pred(30)
	S5	WF, DA	–
	S7	RA	R-squared
	S12	RA	–
	S17	CBR	–
	S78, S80, S82	Statistical	–

Accuracy criteria acronyms: Magnitude of Relative Error (MRE), Mean MRE (MMRE), quartiles of MRE distribution (qMRE), Standard Deviation of MRE (Std.MRE), Coefficient of correlation R, Coefficient of determination (R-squared), Percentage Relative Error Deviation (Pred(0.25), Pred(0.30), Pred(0.75)), Mean Absolute Error (MAE), Minimum AE (MinAE), Maximum AE (MaxAE), Coefficient of correlation (R), Root Mean Square Error (RMSE), Normalized RMSE (NRMSE), Standard Error of the Estimate (Std.EE), Weighted Average Precision (WAP), Area Under Curve (AUC), True Positive Rate (TPR), False Positive Rate (FPR), Weighted Average Recall (WARec), Spearman's coefficient of correlation (Rs), Cut-off factors (cut-off), Mean Square Error (MSE), Mean Absolute Relative Error (MARE), Roc Area (ROC), Sum of Absolute Residual (Sum Ab. Res), Standard Deviation of Absolute Residual (Std. Ab. Res), Median of Absolute Residual (Med. Ab. Res), Standard Error of the Mean (SEM). Area Over Curve (AOC).

e-Informatica Software Engineering Journal, Volume 13, Issue 1, 2019, pages: 203–226, DOI 10.5277/e-Inf190106

Measuring Goal-Oriented Requirements Language Actor Stability

Jameleddine Hassine*, Mohammad Alshayeb*
Information and Computer Science Department, King Fahd University of Petroleum and Minerals
jhassine@kfupm.edu.sa, Alshayeb@kfupm.edu.sa

Abstract

Background: Goal models describe interests, preferences, intentions, desired goals and strategies of intervening stakeholders during the early requirements engineering stage. When capturing the requirements of real-world systems such as socio-technical systems, the produced goal models evolve quickly to become large and complex. Hence, gaining a sufficient level of understanding of such goal models, to perform maintenance tasks, becomes more challenging. Metric-based approaches have shown good potential in improving software designs and making them more understandable and easier to maintain.
Aim: In this paper, we propose a novel metric to measure GRL (Goal-oriented Requirements Language) "actor stability" that provides a quantitative indicator of the actor maintainability.
Method: We first, validate the proposed metric theoretically then empirically using a case study of a GRL model describing the fostering of the relationship between the university and its alumni.
Results: The proposed actor stability metric is found to have significant negative correlation with the maintenance effort of GRL models.
Conclusions: Our results show that the proposed metric is a good indicator of GRL actors' stability.

Keywords: Goal models, Goal-oriented Requirements Language (GRL), stability, metrics, maintenance

1. Introduction

Software systems must evolve to meet customer needs, business environment, technologies and regulations. Several studies have shown that requirements evolution can significantly affect overall project costs and schedule [1, 2]. Requirements evolution management has emerged as one important topic in requirements engineering research [3]. ISO/IEC 25010 [4] specified eight characteristics for software product external quality, one of which is maintainability which contains modifiability is a sub-characteristic. Modifiability is a combination of changeability and stability. Stability is the ability of the software to remain stable when modified. Maintainable software tends to have a better estimation of the change cost and better prediction of the resulting quality [1].

Goal models are used in order to make sure that stakeholders' interests and priorities are met in the early requirements engineering stages. Goal modeling is an effective approach to represent and reason about stakeholders' goals using models. Over the past two decades, numerous goal-oriented modeling languages and approaches have been introduced (e.g. i* [5], NFR Framework [6], Keep All Objects Satisfied (KAOS) [7], TROPOS [8] and the Goal-oriented Requirements Language (GRL) [9] part of the ITU-T standard User Requirements Notation (URN)). In addition, there were few attempts to propose domain-specific languages, such as DSML/GoalML [10], DSL/KAOS [11] and ARMOR/KAOS [12].

As goal models grow in size and complexity (e.g. large socio-technical systems having many interdependent stakeholders), they be-

Submitted: 11 November 2018; Revised: 26 September 2019; Accepted: 26 September 2019; Available online: 30 October 2019

come difficult to maintain. To address this challenge, numerous goal-oriented metrics-based techniques [13–18] have been proposed. These techniques vary in their targeted notation, their aim, their selected analysis (e.g. quantitative, qualitative, hybrid) and their targeted scope (e.g. global (targeting the entire goal model), local (focusing on one specific actor or path)).

Goals are known to be much more stable than requirements [19, 20]. More specifically, the higher level the goal is, the more stable it is [19]. However, in a fast-changing world, goal models are deemed to evolve to meet constant changes of business needs and stakeholders' intentions. A goal model requires maintenance when there is a shift in stakeholder's motivations, e.g. adoption of new goals or ceasing to support existing goals. Goals may become undesirable or infeasible to realize, e.g. goals might become too costly to realize or non-compliant with new regulations [21]. Hence, an outdated representation of stakeholders' intentions can easily lead to systems that do not fulfill their purpose. According to a recent survey by Horkoff et al. [22], interest in adaptation/variability/evolution of goal models has increased recently compared to other goal-oriented requirements engineering (GORE) topics. Although, there is a variety of empirical evaluations in the area of modeling languages in general (assessing different qualities, e.g. syntactic, semantic, pragmatic, completeness, comprehensibility, complexity, etc.), the majority of the studies in GORE [23] focus on providing empirical evidence of the applicability of goal-oriented notations for specific domains [24], such as collaborative systems [25], socio-technical systems [26] and knowledge transfer [27].

Most of the existing work that addresses the evolution of goal models, focuses mainly on handling inconsistencies (such as tolerating, diagnosing and tracking inconsistencies) [28–31] and modeling and analysis of evolution over time [32, 33]. However, both approaches introduced in [32] and [33] consider only the evolution of goals' satisfactions values (qualitative and quantitative) and do not discuss the evolution of the goal model structure.

Measuring stability provides better estimation of the cost and effort and better prediction of the software quality [34]. Instable software tends to increase maintenance cost; in some cases, the maintenance cost may reach up to 75% of the software total cost [1, 2]. Stable software, on the other hand, reduces maintenance cost. We believe that the design of a GRL actor stability metric would provide information about GRL actors and their evolution; which will provide control over actor-specific change amplification. Furthermore, measuring GRL actor stability gives an indicator of the GRL actor and model maintainability since stability is directly related to maintainability [4].

The main motivation of this paper is to propose a metric to support the maintainability of goal models during the requirements modeling and analysis phase. In particular, we focus on measuring quantitatively the stability of actors across many versions of the goal model. This paper provides the following contributions:

– Propose a novel metric to measure GRL actor stability. To the best of our knowledge, no goal-based stability metrics were introduced in the literature.
– Validate theoretically and empirically the proposed GRL-based actor stability metric.
– Provide a foundation for systematic assessment of GRL actor stability with respect to the many changes undergone by GRL models during the development life cycle, e.g. model refinement, validation, and maintenance.

The remainder of this paper is organized as follows. In Section 2, we review the current state of the art. Section 3 introduces briefly the GRL language. Our proposed actor stability metric is presented in Section 4. In Section 5, we provide theoretical and empirical validation of the proposed metric and we discuss the possible threats to validity. Section 6 discusses the interpretation and benefits of the proposed metrics. Finally, conclusions and future work are presented in Section 7.

2. Related work

In this section, we review software stability metrics, goal models and requirements stability measurements.

2.1. Software stability

Researchers proposed stability metrics at system [35–39], model [40], architecture [41–44] and class levels [38, 45–47]. Classes in object-oriented (OO) systems form the basic building blocks and thus they are the most related stability metrics to the metric proposed in this paper as we propose a stability metric at actor level, which is also the basic building block for GRL models, hence, we discuss these metrics in details.

Li et al. [38] proposed the "Class Implementation Instability"(CII) metric to measure the evolutionary change in the implementation of a class. The authors in [38] measure class instability by measuring the lines of code added, deleted, or modified between two versions. CII metric is not normalized; therefore, its value has no upper or lower limit. Grosser et al. [45] proposed a metric to measure the class stability based on the method interface; the class is considered stable if the method interfaces are unchanged between versions. Hence, the metric measures the number of methods whose interface (signature) has not been changed between two versions regardless of the changes occurred to the method bodies. According to Grosser et al. [45], the class is fully stable when all method signatures available in one version are available in the other version. On the other hand, the class is considered fully instable, when none of the method signatures remain unchanged between the two versions. This metric is normalized and yields a value between 0 and 1. Ratiu et al. [46] proposed a class stability metric that uses the number of methods in a class between two versions. According to Ratiu et al. [46] the class can be either stable or instable, i.e. the value of the metric can be either 0 for instable or 1 for stable classes. A class is stable when the number of methods between two versions remain is unchanged; the class is instable when there is a change in the number of methods between two versions. Alshayeb et al. [47] proposed a Class Stability Metric (CSM) to measure Object-Oriented (OO) class stability. The authors [47] analyzed the OO class properties and identified eight class properties that affect class stability. These properties are: method access-level, method code, method signature, class variable access-level, class variable, class access-level, class interface name and inherited class name. For each property, the authors [47] measure the extent of change between two versions by measuring the unchanged properties. The class stability is measured by aggregating the individual stability values for all the properties. CSM is normalized and hence the value of the metric can be between 0 (fully instable) and 1 (fully stable).

There are three approaches to defining software systems' stability for model or code levels. The first approach is that the software system is stable if no changes are made to the software artifact being measured. Thus, the software is fully stable when the original and the subsequent version are identical [48]. The second approach considers the software system as stable if it avoids addition of new artifacts or modification of existing ones, thus deletion is considered as modification [34]. The third approach allows additions to the existing software system. Thus, the software is considered fully stable when there are no changes to the existing artifacts regardless of the additions that might be made to the system [49]. In this paper, we adopt the third stability definition when defining the proposed metric.

2.2. Goal models measurement

There is a growing body of literature on goal-oriented metrics [5, 7, 13–16, 18, 50–52]. Kaiya et al. [17] proposed quality metrics (introduced as part of the AGORA (Attributed Goal-oriented Requirements Analysis) approach) to measure correctness, unambiguity, consistency, verifiability, modifiability, traceability, and completeness of AND-OR goal graphs according to a stakeholder preference matrix. The proposed metrics are global and do not consider dependencies between intervening actors (used to describe stakeholders and systems in goal models). Franch and Maiden [13] introduced metrics to quantify i* Strategic Dependencies (SD) [5] models, that can be used to help choosing the most appropriate Commercial Off-The-Shelf (COTS) components.

These quantitative metrics are based on a categorization of the different types of strategic dependencies into duplicated and non-duplicated, hidden and non-hidden, resource and non-resource, etc. The resulting metrics are then applied to measure six system properties, namely, Diversity, Vulnerability, Packaging, Self-Containment, Uniformity, and Connectivity. In Franch et al. [14], i* SD actors and dependencies are categorized into sorts, e.g. human/computer, goal/task, etc. The proposed framework [14] implements three structural metrics, aiming to assess system properties, such as privacy, accuracy and efficiency. The proposed framework supports both global and local metrics and takes into account actor and dependency weights (i.e. importance values). Later, the approaches introduced in [13] and [14], have been applied by Grau et al. [16, 18] to assess the effectiveness of alternative architectures. In order to evaluate an architectural property, the authors proposed a coupling metric over i* SD models. Coupling is measured by the number of incoming and outgoing dependencies (multiplied by a weight factor relative to each actor) an actor is associated with. To measure model predictability, Franch [15] has proposed a framework that considers both i* SD and SR (Strategic Rationale) models. The predictability metric, expressed in Object Constraint Language (OCL), can be local or global and may require expert judgment. More recently, Franch [51] has proposed a method based on system domain analysis for defining metrics in i* using the iMDF framework. Espada et al. [52] proposed quantitative metrics for evaluating the complexity and the completeness of KAOS [7] goal models. The authors used the Goal-Question-Metric (GQM) approach [50]. However, their approach does not consider dependencies between KAOS agents. Gralha et al. [53] proposed a set of metrics to measure and analyze complexity and completeness of goal models. In the GRL [9] context, Hassine and Alshayeb [54], proposed a structural metric to measure actor external dependencies (AED). Furthermore, jUCMNav [55] tool (GRL modeling and analysis framework) captures many simple structural GRL metrics (e.g. number of actors, goals, tasks, intentional elements, intentional links, etc.) and allows for the definition of additional metrics using OCL.

In this paper, we extend the set of existing GRL metrics [54, 55] by introducing a novel metric to measure GRL actor stability. Our proposed metric is (1) structural in the sense that it depends only on the connectivity of a given GRL model and not on its semantic, (2) local since it is applied at the actor level rather than the entire GRL model, and (3) quantitative since it measures the degree of actor stability with respect to changes a GRL model can undergo, e.g. model refinement and maintenance.

2.3. Requirements stability measurement

Many metrics have been proposed to understand the sources, frequencies and types of requirements evolution. Lam and Shankararaman [56] proposed a change volatility metric to measure the number or proportion of changes, within a specified period. Change volatility [56] helps assess the stability of a system. A high-risk requirement may be characterized by a high-level change volatility. Anderson and Felici [57] proposed the Requirements Maturity Index (*RMI*), a metric used to quantify the readiness of requirements. The *RMI* is computed as follows:

$$RMI = \frac{RT - RC}{RT}$$

where *RT* is the total number of software requirements in the current release and *RC* is the number of requirements changes, i.e. added, deleted or modified requirements, allocated to the current release. It is worth noting that *RMI* metric is sensitive to requirements change in successive releases, but it does not take into account historical information about change. To address this issue Anderson and Felici [58] refined *RMI* by introducing the Requirements Stability Index (*RSI*) (a metric used to measure the extent of requirements stability and the frequency of changes to requirements) and the Historical Requirements Maturity Index (*HRMI*). *RSI* takes into account *CRC* (the cumulative number of requirements changes) and is defined as follows:

$$RSI = \frac{RT - CRC}{RT}$$

HRMI takes into consideration *ARC* (the average number of requirement changes) and is defined as follows:

$$HRMI = \frac{RT - ARC}{RT}$$

The authors [57] claimed that *HRMI* are less sensitive than *RMI* to changes over consecutive releases. Stark et al. [58] characterized requirements volatility as additions to the delivery content, deletions from the delivery content and changes in scope to an agreed-upon requirement. More recently, AbuHassan and Alshayeb [40] proposed a suite of stability metrics for UML use case models, UML sequence diagrams and UML class diagrams. However, in the context of goal-oriented languages and to the best of our knowledge, no stability metrics have been proposed. In this paper, we aim to fill this gap by proposing a novel metric to measure GRL "actor stability" that provides a quantitative indicator of the actor maintainability, allowing for a better estimation of GRL models change.

3. GRL in a nutshell

The Goal-oriented Requirements Language (GRL) [9] is an ITU-T standard visual goal modeling language used to model stakeholders' intentions, business goals and non-functional requirements. GRL is based on i* [5] and the NFR framework [6]. In what follows, we briefly introduce the different GRL constructs.

3.1. GRL actors

An actor (illustrated as), where the name of the actor reference is shown as a label next to a stickman icon on the top-left side of the dashed ellipse) represents an entity that has intentions and carries out actions to achieve its goals by exercising its know-how. Actors are often used to represent stakeholders as well as systems.

3.2. GRL intentional elements and indicators

There are five different types of intentional elements:

1. *Goal* (illustrated as): A (hard) *Goal* (either a business goal or a system goal) is a condition or state of affairs in the world that the stakeholders would like to achieve.
2. *Softgoal* (illustrated as): is a condition or state of affairs in the world that the actor would like to achieve, but there are no clear-cut criteria for whether the condition can be entirely achieved. However, it can be sufficiently achieved. Softgoals are often used to describe non-functional aspects such as availability, security, etc.
3. *Task* (illustrated as): states a particular way of performing something. Tasks can be considered as the operations, processes, data representations, structuring and constraints used to meet the needs stated in the goals and softgoals of the target system.
4. *Resource* (illustrated as): is a physical or informational entity.
5. *Belief* (illustrated as): used to represent design rationale. Intentional elements may be included in actor definitions and they can be linked to each other in different ways. In addition to intentional elements, GRL defines *indicators* (illustrated as) to describe a qualitative or quantitative real-world measurement.

3.3. GRL links

There are five types of GRL links [9]:

1. *Contributions* (illustrated as $\longrightarrow$): describe how a source intentional element or source indicator contributes to the satisfaction of a destination intentional element. A contribution has a qualitative level and an optional quantitative value.
2. *Correlations* (illustrated as $\dashrightarrow$): express knowledge about interactions between inten-

tional elements. A correlation link is similar to a contribution link except that the correlation is not an explicit desire but is a side-effect and that correlations are only used with intentional elements and not with indicators.

3. *Dependencies* (illustrated as ─▸─): enable reasoning about how actor definitions depend on each other to achieve their desired goals. It describes how a source actor (the *depender*) depends on a destination actor (the *dependee*).
4. *Decompositions* (illustrated as ──┼): provide the ability to define what source intentional elements need to be satisfied in order for a target intentional element to be satisfied. There is no ordering between the decomposing elements. A decomposition link can be one of the following:
   – *AND* decomposition: The satisfaction of each of the sub-intentional elements is necessary to achieve the target.
   – *XOR* decomposition: The satisfaction of one and only one of the sub-intentional elements is necessary to achieve the target.
   – *OR* decomposition: The satisfaction of one of the sub-intentional elements is sufficient to achieve the target, but many sub-intentional elements can be satisfied.
5. *Belief links* (illustrated as ----): used to connect beliefs to GRL intentional elements.

3.4. Qualitative contributions

The qualitative contribution of a source intentional element or indicator to a destination intentional element can be one of the following values based on the degree (positive or negative) and sufficiency of the contribution to the satisfaction of the destination intentional element:

1. *Make* (illustrated as $\dot{+}$): the contribution is positive and sufficient.
2. *Help* (illustrated as $\bar{+}$): the contribution is positive but not sufficient.
3. *SomePositive* (illustrated as +): the contribution is positive, but the extent of the contribution is unknown.
4. *Unknown* (no symbol on the link): there is some contribution, but the extent and the degree (positive or negative) of the contribution is unknown.
5. *SomeNegative* (illustrated as −): the contribution is negative, but the extent of the contribution is unknown.
6. *Hurt* (illustrated as $\underset{\cdot}{-}$): the contribution is negative but not sufficient.
7. *Break* (illustrated as $\dot{-}$): the contribution of the contributing element is negative and sufficient.

It is worth noting that GRL is permissive in how intentional elements can be linked to each other, contrary to i* [5] which imposes restrictive usage of relationships (e.g. a contribution link cannot have a task as a destination).

For a detailed description of the GRL language, the reader is invited to consult the URN (User Requirements Notation) ITU-T standard [9].

3.5. GRL example

Figure 1 illustrates a GRL model composed of one GRL actor, called Commuter, who wants to minimize the time lost during a commute (modeled as a GRL goal "Minimize time lost by commute"). Two goals "Work during commute" and "Minimize travel time" contribute positively (through two *Help* contributions) to the achievement of the upper goal. While taking public transportation (represented as a goal called "take public transportation") contributes positively (through a *Help* contribution) to the achievement of the goal "Work during commute", it contributes negatively (through a *Hurt* contribution) to the achievement of the goal "Minimize travel time". Similarly, taking private transportation (represented as a goal called "take private transportation") contributes positively (through a *Help* contribution) to the achievement of the goal "Minimize travel time", it contributes negatively (through a *Hurt* contribution) to the achievement of the goal "Work during commute". When it comes to use public transportation, the commuter has the choice (illustrated as an OR decomposition link) between taking the regular bus (i.e. illustrated as task "Take regular bus") or taking the express bus (i.e. illustrated as task "Take express bus"). Furthermore, two options (illustrated as an OR decomposition

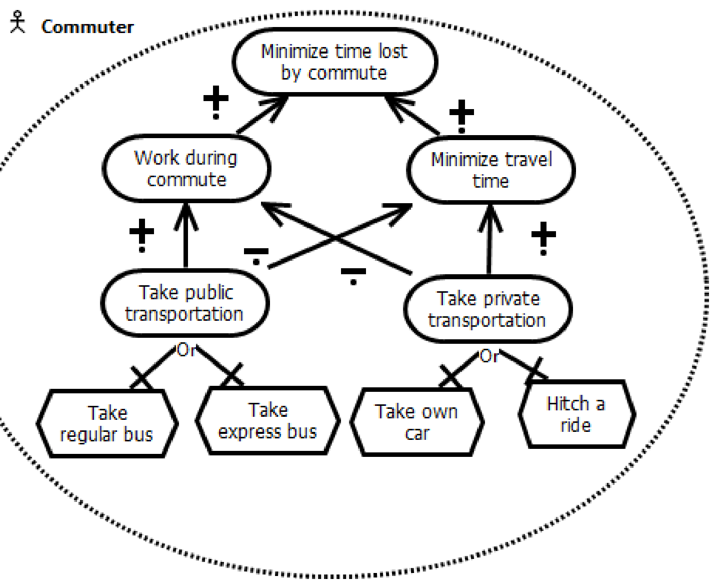


Figure 1. GRL commuter example

link) are available to the commuter when it comes to the use of the private transportation: take his own car (represented as task "Take own car") or hitch a ride (represented as task "Hitch a ride").

4. Measuring GRL actor stability

In this section, we propose a novel metric to measure GRL actor stability and we provide an example to illustrate its calculation.

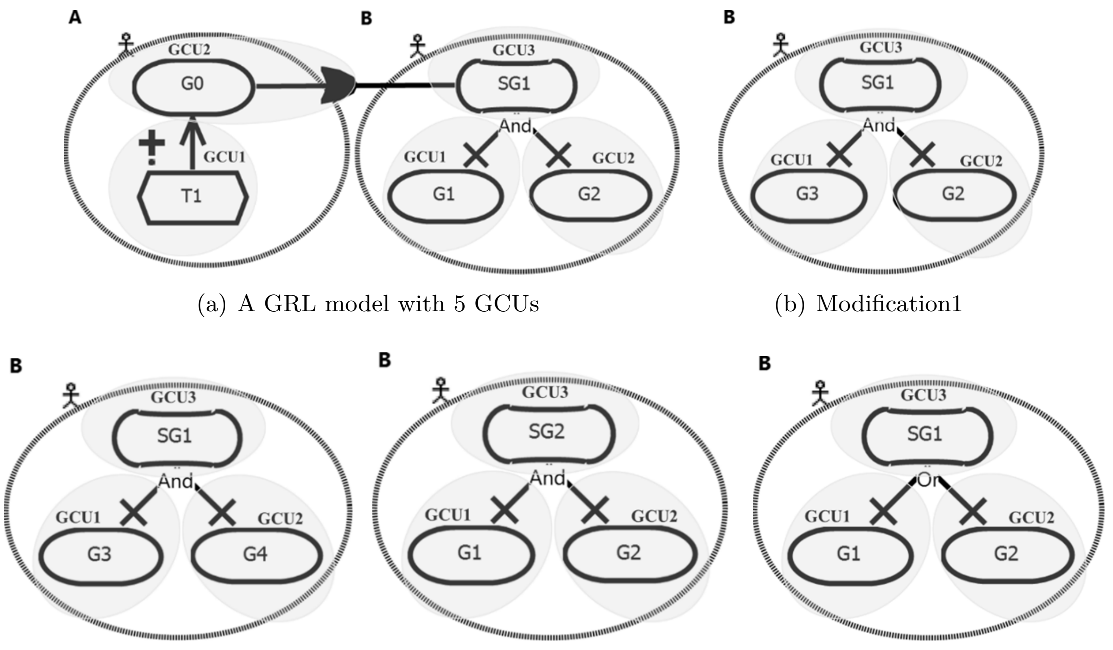
4.1. GRL Change Unit (GCU)

Conducting a maintenance task on a GRL model generates a new version (version i is the current version and version $i + 1$ is the modified version). To quantitatively assess the magnitude of a change, we should characterize the basic units (GRL sub-models) that are subject to change. We define the "GRL Change Unit (GCU)", as being:

1. An intentional element combined with its outgoing link (with or without a quantitative value). It is worth noting that a GCU may have an outgoing link that crosses the boundary of the containing actor to reach another intentional element contained within another actor.
   Or
2. An intentional element that does not have any outgoing links.

Links are not unique while intentional elements are unique within an actor, therefore, many similar links may exist in the actor model and hence we will not be able to identify which links remain unchanged. Therefore, we combine the intentional element with its outgoing link to form a GCU. Figure 2a illustrates a generic GRL example composed of two actors A and B. Actor A is composed of two GCUs: (1) GCU1 composed of task T1 and a *help* contribution and (2) GCU2 composed of goal G0 and the dependency link. Actor B is composed of three GCUs: (1) GCU1 composed of goal G1 and the AND decomposition, (2) GCU2 composed of goal G2 and the AND decomposition and (3) GCU3 composed of the softgoal SG1.

Compared with version i of a GRL model, version $i + 1$ may have added, deleted, changed, and unchanged GCUs. It is worth noting that we consider both syntactic and semantic changes. Examples of possible changes include: changing the type of an intentional element (e.g. from a goal to a task), changing the type of a decomposition (e.g. from OR to AND), changing the qualitative/quantitative values of a contribution (e.g. from *help* to *hurt*), changing the text of an intentional element with a different text not having the same meaning (semantic change), re-



(a) A GRL model with 5 GCUs (b) Modification1

(c) Modification2 (d) Modification3 (e) Modification4

Figure 2. GRL Change Unit (GCU)

arranging the text within an intentional element (although such a change seems to impact only the syntactic aspect, it may also impact the semantics if the text meaning changes). However, fixing typos in the text of an intentional element is not considered as a change. Furthermore, since the GCU represents the smallest change unit, we count for only one single change when either the intentional element changes or the outgoing link changes or both changes.

In what follows, we discuss and justify the GCU definition through some examples of possible changes. Assume that we perform the following changes to the GRL actor B of Figure 2a:

1. **Modification1**: Replace goal G1 by another goal G3 (see Figure 2b). Intuitively, it should be accounted as a single change, which is captured by the GCU definition as a change in GCU1 only (GCU2 and GCU3 remain unchanged). Indeed, since G1 is part of GCU1 only (within actor B), replacing G1 by G3 would only affect GCU1.
2. **Modification2**: Replace goal G1 by goal G3 and replace goal G2 by goal G4 (see Figure 2c). Intuitively, these two replacements should be accounted as two changes, which is captured by the GCU definition as two changes in both GCU1 and GCU2, while GCU3 remains unchanged. Considering the AND-decomposition as a single unit (as opposed to our current definition of GCU) would not reflect the amount of applied change.
3. **Modification3**: Replace the softgoal SG1 by softgoal SG2 (see Figure 2d). Intuitively, it should be accounted as a single change, which is captured by the GCU definition as a change in GCU3 only (GCU1 and GCU2 remain unchanged). The enclosure of the target element into the GCU (as opposed to our current definition of enclosing source and link only), e.g. G1–AND–SG1 and G2–AND–SG1, would lead to double counting the change that impacts SG1.
4. **Modification4**: Replace the AND-decomposition by an OR-decomposition (see Figure 2e). This change has an impact on how goals G1 and G2 contribute to the achievement of SG1.

Hence, both links of the decomposition are modified from AND to OR, which is captured by the GCU definition as two changes in GCU1 and GCU2 (GCU3 remains unchanged). Considering the AND decomposition as a single unit would not reflect the fact that many children have to contribute differently to their parent node.

GRL beliefs are connected to intentional elements through belief links, presenting no specific direction. We assume that a belief link terminates at a GRL belief. Since indicators are used only in converting real-world values into satisfaction levels, they are out of the scope of this research.

4.2. GRL actor stability metric

The objective of this paper is to propose a metric to measure GRL actor stability. We will follow a similar approach to the one by Alshayeb et al. [47]. We will reason about GCU as being the basic unit of change. We define possible changes between versions i and $i+1$ as follows:

1. Added GCU means that the GCU was not present in GRL model version i and it has been added in version $i+1$.
2. Deleted GCU means that the GCU was present in GRL model version i and has been deleted in version $i+1$.
3. Changed GCU means that the GCU was present in GRL model version i and has been changed in version $i+1$.
4. Unchanged GCU means that the GCU was present in GRL model version i and has neither been deleted nor changed in version $i+1$.

All actors within a GRL model are tagged with a version number. When a change occurs in any actor, the new GRL model is tagged with a different version number, even though the model may contain unchanged actor(s). Since we are measuring the stability of the GRL actor, we will measure the number of GCUs that have been unchanged.

We measure the stability between two consecutive versions, i.e. we measure the stability of version $i+1$ with respect to its previous version i. Version $i+1$ is considered fully stable when all GCUs in version i have not been changed or removed and is considered fully instable if all of its GCUs have been changed or removed as compared to version i.

To measure the GRL actor stability, we measure the stability of each GRL intentional element type for the actor and then sum the stabilities of all GRL Intentional element type for that actor. To calculate the stability of each GRL Intentional element type, we use the steps shown in Algorithm 1. Equations (1–5) show the metrics used to calculate each GRL actor intentional element stability.

If the intentional element has more than one outgoing link, each link is treated independently. The actor stability value ranges from 0 to 1, with 0 denoting completely instable and 1 denoting completely stable actor.

To calculate the GRL actor stability of version $i+1$ with respect to version i, we then average the stabilities of all GRL actor intentional element stability as shown in equation (6).

$$\text{Goal Stability(GS)}_{i,i+1} = \frac{\text{number of unchanged Goals between model version } i \text{ and version } i+1}{\text{number of Goals in model version } i} \tag{1}$$

$$\text{Softgoal Stability(SS)}_{i,i+1} = \frac{\text{number of unchanged Softgoals between model version } i \text{ and version } i+1}{\text{number of Softgoals in model version } i} \tag{2}$$

$$\text{Task Stability(TS)}_{i,i+1} = \frac{\text{number of unchanged Tasks between model version } i \text{ and version } i+1}{\text{number of Tasks in model version } i} \tag{3}$$

$$\text{Resource Stability(RS)}_{i,i+1} = \frac{\text{number of unchanged Resources between model version } i \text{ and version } i+1}{\text{number of Resources in model version } i} \tag{4}$$

$$\text{Belief Stability(BS)}_{i,i+1} = \frac{\text{number of unchanged Beliefs between model version } i \text{ and version } i+1}{\text{number of Beliefs in model version } i} \tag{5}$$

$$\text{Actor Stability}_{i,i+1} = \frac{GS + SS + TS + RS + BS}{\text{number of distinct types of intentional elements in model version } i} \tag{6}$$

Algorithm 1: Measuring GRL actor stability

```
Input   : Actor A
Output  : Stability of Actor A
Let T denote the set of distinct intentional element types {GS, SS, TS, RS, BS} within Actor A
foreach $t \in T$ do
    Compute unchangedCount;
    ▷ Count the number of unchanged GCUs (the number of unchanged occurrences of
      that Intentional element type between versions i and i+1
    Compute maximumPossibleChangeCount;
    ▷ Count the maximum possible change of that GCU (the number of occurrences of
      that Intentional element that exists in version i)

    $extentOfChange[t]_{(i,i+1)} = \frac{UnchangedCount_{(i,i+1)}}{maximumPossibleChangeCount \text{ in model version } i};$
end
▷ Compute Actor's A stability

$ActorStability_{(i,i+1)} = \frac{\sum_t extentOfChange[t]_{(i,i+1)}}{\text{number of distinct intentional element types in model version}i};$

return ActorStability;
```

Where 1 (at least one type) $\leq$ number of distinct types of intentional elements $\leq$ 5 (goals, softgoals, tasks, resources and beliefs).

The proposed metric neither measures the percentage of change nor the number of changes, it rather measures the extent of change between two versions. The extent of change is measured by aggregating the individual stability values for all the intentional elements types. A GRL actor stability value of $x\%$ does not mean $x\%$ of the actor elements have unchanged, rather, it means that $x\%$ of the actor model structure remains unchanged.

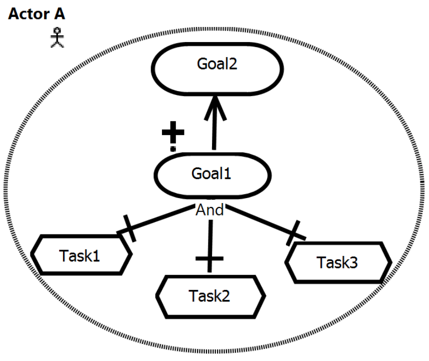
To reduce the impact of having one change in simple actor that has few possible changes as compared to a single change for a complex actor, we do not just calculate the number of possible changes for all GCUs in the actor together; rather, we consider the actor to have 5 main intentional element types and we calculate the extent of change for each intentional element type separately. Thus, the change will be with respect to the GCUs for that particular intentional element type.

4.3. Measuring GRL actor stability example

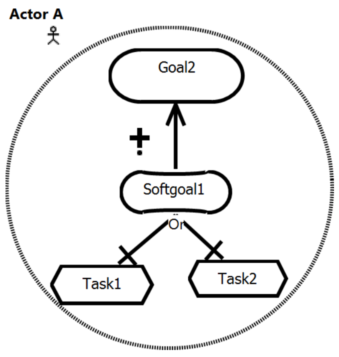
In this section, we apply the proposed actor stability metric to a generic GRL example. Figure 3a illustrates an AND decomposition within actor A. In Figure 3b, the AND decomposition is converted to an OR decomposition, the goal *Goal1* is changed to *Softgoal1* and *Task3* is deleted.

To calculate the extent of change, we first calculate the *maximumPossibleChangeCount* as proposed in [59]. The *maximumPossibleChange-Count* counts the maximum possible change that can happen to each property with respect to version i, thus it measures the number of GCUs for each property that exists in the model, as shown in Table 1. For example, in Figure 3a, we have two GCUs with goals as intentional elements: (1) *Goal1* with the *help* link and (2) *Goal2*. In addition, we have three GCUs with tasks as intentional elements: (1) *Task1* with the AND decomposition link, (2) *Task2* with the AND decomposition link and (3) *Task3* with the AND decomposition link. The remaining types of intentional elements (i.e. Softgoal, Resource, Belief) are not present in the model. Hence, their maximum possible change is zero.

Then, calculate the number of GCUs that have not been changed between the two versions as given in Table 2. In Figure 3b, the GCU composed of Goal2 has not been changed, while the GCU composed of *Goal1* and the *help* contribution link has been changed, re-



(a) Initial decomposition



(b) Changed decomposition

Figure 3. GRL example

Table 1. Calculate maximum-possible-change for each GCU' intentional element

GCU intentional element	Maximum possible change
Goals	2
Softgoals	0
Tasks	3
Resources	0
Beliefs	0

sulting into *Softgoal1* and a *help* contribution link. Hence, the number of unchanged GCUs with a goal as intentional element is 1. Since, the AND decomposition is changed to an OR decomposition, the three GCUs involving tasks as intentional elements are considered as changed. Consequently, their number of unchanged GCUs is 0.

Finally, calculate stability value for each property and the whole GRL actor stability as shown in Table 3. This is done by dividing number of unchanged GCU by maximum possible change for each property. The GRL actor stability is computed as the total stability divided by the number of distinct intentional element types involved in the model, since not all intentional element types might be present in the model. In this example, it is 2 (Goals and Tasks).

5. GRL actor stability metric validation

Theoretical and empirical validation of software metrics are usually performed before they can be used with confidence. In this section, we validate the proposed GRL actors stability metric theoretically and we evaluate it empirically.

5.1. Theoretical validation

Theoretical validation refers to the process of certifying that the metric confirms to the principles of measurement theory. Different frameworks have been proposed to validate software metrics [60–63]. However, no framework has been found to specifically validate stability metrics. Therefore, we use Kitchenham et al. framework [64] which is a generic metric validation framework, to theoretically validate the proposed GRL actor stability metric. They proposed a framework for validating software measurement; the framework contains four properties the metric should satisfy to be theoretically valid. These properties are: (1) different entities must be distinguished from each other, (2) the valid measure must satisfy the representation condition, (3) units that contribute to the valid measure must be equivalent and (4) different entities can have the same attribute value. In addition to Kitchenham's et al. framework [64], we show the validation through an example. Figure 4 shows an

Table 2. Calculation of the number of Unchanged GCU' intentional wlements

GCU intentional element	Number of unchanged GCUs
Goals	1
Softgoals	0
Tasks	0
Resources	0
Beliefs	0

Table 3. Stability calculation for each property and the overall actor stability

GCU intentional element	Maximum possible change	Number of nnchanged GCUs	Stability
Goals	2	1	0.5
Softgoals	–	–	–
Tasks	3	0	0
Resources	–	–	–
Beliefs	–	–	–
Total stability = 0.5 + 0 = 0.5			
GRL actor stability = 0.5/2 = 0.25			
Note: "–" means the intentional element is not present in the actor.			

example of four versions of a GRL actor enclosed elements, the base version (version 0) and three other versions (versions 1 to 3). The examples are used to demonstrate the validity of the proposed metric against Kitchenham's et al. theoretical validation properties. For simplicity, we only use goals in these examples; however, what applies to goals applies to the other intentional elements.

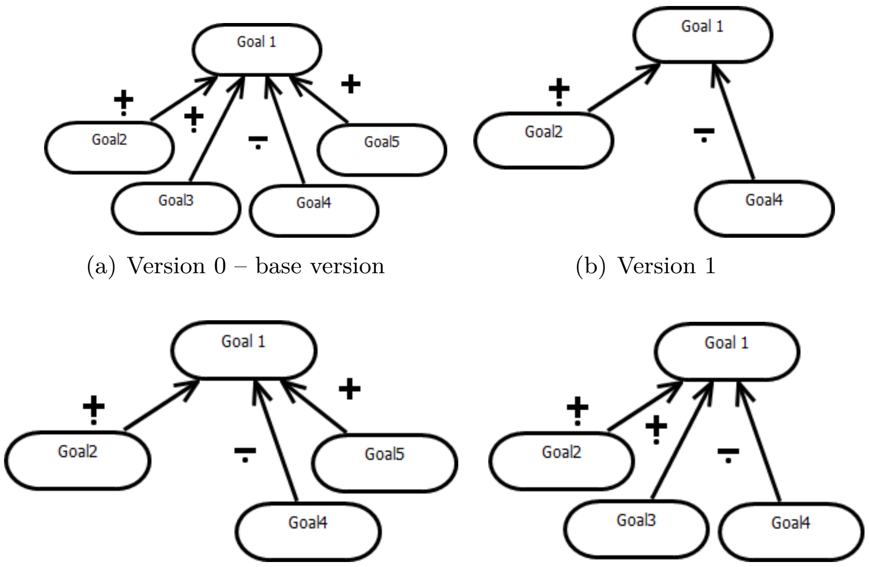
Property 1: "For an attribute to be measurable, it must allow different entities to be distinguished from one another" [64]. That is, different entities should have different measurement values, thus, the stability value for two actors will be different if they have different number of unchanged entities between the two versions.

To validate this property, consider the GRL models in Figures 4a–4c. As compared to the base version of Figure 4a, in Figure 4b three goals remained unchanged (*Goal1*, *Goal2*, and *Goal4*) while in Figure 4c four goals remained unchanged. Since the number of unchanged goals in Figure 4b and Figure 4c with respect to the base version Figure 4a are different, stability values should also be different. This will always be true as the denominator will be the same when calculating the stability for both versions (number of goals in the base version) while the numerator value (unchanged goals) will be different; therefore, the overall stability value will also be different. By calculating the stability value for these versions, we notice that the stability value for the GRL model shown in Figure 4b is 0.6 while the stability value for the GRL model shown in Figure 4c is 0.8. Thus, this property is shown to be true.

Property 2: "A valid measure must obey the Representation Condition" [64]. That is, if more entities have been unchanged between the two versions in two GRL actors, then the stability value of the GRL actor that has less unchanged entities should be higher.

To validate this property through the example, consider the same scenario used to prove property 1. The GRL model shown in Figure 4b has less unchanged properties as compared to the GRL model shown in Figure 4c; this is reflected in their stability values being 0.6 and 0.8, respectively. This property will always be true as the value in the denominator will be the same when calculating the stability for both versions while the numerator value will be different and when more unchanged properties exists, the numerator value will be higher and thus the overall stability is higher.

Property 3: "Each unit of an attribute contributing to a valid measure is equivalent" [64]. Consider Figure 4b and Figure 4c, in the same line



(a) Version 0 – base version (b) Version 1

(c) Version 2 (d) Version 3

Figure 4. Different versions of GRL models

as property 1 and property 2, the denominator will be the same when calculating the stability for both versions (five in the example shown in Figure 4). Each change in the goal will have the same weight since it has impact of 0.2 (which is 1/5); thus, each change contributes by the same weight. In Figure 4b, three goals are unchanged, which makes its stability 0.6; there are four unchanged goals in Figure 4c, which makes its stability 0.8. Since the three goals still exist in Figure 4c and one more goal has unchanged, the stability of the model in Figure 4c should be equal to the value of actor stability in Figure 4b (0.6) plus the stability of the individual goal that has unchanged (0.2), which is shown to be true as the total stability of the model in Figure 4c is 0.8.

Property 4: "Different entities can have the same attribute value" [64]. That is, two GRL actors can have the same stability value if the same number of GCUs have unchanged when they have the same number of GCUs in each distinct intentional element.

Finally, to show the validity of property 4, consider the GRL models shown in Figure 4c and Figure 4d. Figure 4c has four goals that are unchanged (goal 1, 2, 4 and 5), while Figure 4d has four unchanged goals (goal 1, 2, 3 and 4). We notice that in Figure 4c goal 3 has been deleted, while in Figure 4d goal 5 has been deleted as compared to the base version shown in Figure 4a. Therefore, the two GRL models are different, yet, they both have the same stability value (0.8) when compared to the base version (shown in Figure 4a). This is true because the denominator (number of goals in the base version) is the same in both cases and the numerator (number of unchanged goals) is also the same as the count of unchanged goals is equal regardless of which goals have changed. Therefore, the four properties proposed by Kitchenham et al. [64] are satisfied, thus, the proposed metric is theoretically valid.

5.2. Empirical validation

The empirical validation of a metric helps in assessing its usefulness and relevance. This section describes the experiments carried out to provide empirical evidence with respect to the usefulness and relevance of the proposed actor stability metric. This is achieved by following the templates and recommendations presented in Wohlin et al. [65].

5.2.1. Experiment goals

The main goal of our empirical study is to investigate the relationship between maintainability and the proposed stability metric. If such relationship is revealed by the experiment, then it can be shown that the proposed stability metric can be used as an indicator of the maintenance effort for the GRL actor model. Previous studies used time and effort to measure maintainability. Time is measured by the number of hours spent on maintenance activities [66, 67] and effort is measured by the number of lines added, deleted, or changed [68, 69]. The proposed metric is at the model level; thus, we measure maintainability effort using the time spent on performing the maintenance task. Since the proposed stability metric is measured between two consecutive versions (Stability$(i, i+1)$), we measure the effort (Effort$(i, i+1)$) to produce version $i+1$ using version i as base version.

5.2.2. Experimental design

Empirical studies are conducted to test a theory to provide further evidence to support or reject it [70]. Since software stability is directly related to maintainability [4], we expect that a decrease in software stability will translate to more time spent on maintainability. To empirically validate the proposed metric, we designed and conducted a controlled experiment to test this assumption. In the experiment, we correlate the proposed metric values with the time spent on performing four maintenance tasks. We expect that the more stable an actor is, the less the time required for its maintenance will be. If such relation is observed, we can conclude that the proposed metric is empirically valid. Figure 5 illustrates the main steps of our experimental plan.

1. **Subjects.** Our subjects are 28 undergraduate software engineering students (randomly assigned to 7 groups of 4 members each) and 9 individual software engineering undergraduate students, enrolled in requirements engineering course. This would allow for more variability to gain more confidence in the experiment results. All participants received around 9 hours of training on GRL including hands-on using the jUCMNav tool [55].
2. **Material.** The material given to the subjects consists of printouts of a GRL model that describes how to foster the relationship between a university and its alum.ni. Figure 6 shows the initial version of the designed GRL model that has been adapted from [71].
   The four maintenance tasks to be executed on the GRL model are detailed in Section 5. To address the variability in the experiment, we considered four different actors with different sizes, performed the maintenance tasks on different actors and model constructs and applied all types of changes (modification, addition, and deletion).
3. **Variables.** We measure maintainability by means of the following dependent variables: (1) the time spent by the subjects in performing the four maintenance tasks (in seconds) and (2) the stability values of the four actors computed by the authors after each maintenance task. The independent variable is the performed maintenance tasks.
4. **Hypothesis.** The experiment was planned with the purpose of testing the following hypothesis:
   Null hypothesis (H0): there is no correlation between actor stability and maintainability measured by time spend on performing the maintenance task.
   Alternative hypothesis (H1): there is a correlation between actor stability and maintainability measured by time spend on performing the maintenance task.
5. **Experimental tasks.** The subjects were asked to conduct 4 corrective maintenance tasks. We have considered the following aspects when designing the maintenance tasks: (1) Tasks are small enough so they can be performed by students within a short period of time, (2) Maintenance tasks are not trivial and require careful analysis (to mimic real maintenance tasks), (3) Tasks are not too restrictive. Hence, more than one solution may be retained.
   The four maintenance tasks are as follows:
   – **Maintenance Task 1**: The "Alumni Department" investigated ways to assess precisely how the department can serve alumni. It turned out that the goal "Serving alumni through University commitment" has no clear-cut satisfaction crite-

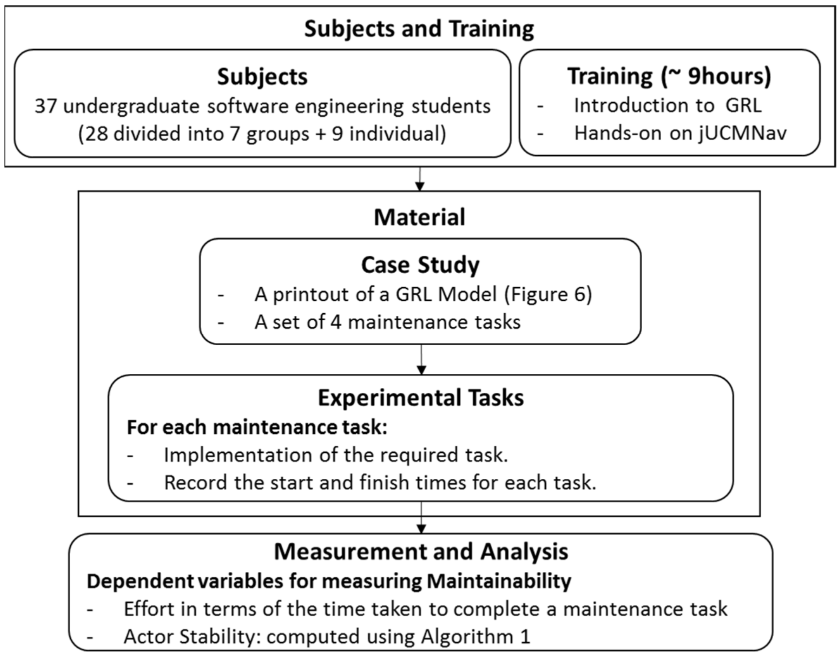

Subjects and Training
Subjects
37 undergraduate software engineering students
(28 divided into 7 groups + 9 individual)
Training (~ 9hours)
- Introduction to GRL
- Hands-on on jUCMNav
Material
Case Study
- A printout of a GRL Model (Figure 6)
- A set of 4 maintenance tasks
Experimental Tasks
For each maintenance task:
- Implementation of the required task.
- Record the start and finish times for each task.
Measurement and Analysis
Dependent variables for measuring Maintainability
- Effort in terms of the time taken to complete a maintenance task
- Actor Stability: computed using Algorithm 1

Figure 5. Experimental design

ria. In addition, the alumnus found that the "Alumni Department" is breaching their privacy by sending too many SMSs. Please fix the GRL model to resolve these two issues.

- **Maintenance Task 2**: Please use the GRL model that you have already modified in maintenance Task 1. The university allocated funds to the alumni department have been reduced. The alumni department has to cope with this constraint by reducing their expenses without affecting the offered activities. Please modify the GRL model to implement these constraints.
- **Maintenance Task 3**: Please use the GRL model that you have already modified in maintenance Task 2. Each semester, the "Alumni Department" has been asking their alumnus to mentor an increasing number of undergraduate students, in their respective fields. However, based on our undergraduate students' feedback, this experience has many shortcomings and was not that positive. According to the alumni, mentoring a large number of students is not sustainable. Please modify the GRL model to reflect this fact.
- **Maintenance Task 4**: Please use the GRL model that you have already modified in maintenance Task 3. Alumni are willing to contribute to the university activities, but they are reluctant to donate money. Please modify the GRL model to reflect this fact.

It is worth noting that for tasks 2, 3, and 4, subjects were asked to not count the time taken to copy changes made in the previous task.

5.2.3. Experiment execution and data collection

In this section, we present samples of the executions of the maintenance tasks along with their corresponding actor stability computation. It is worth noting that some data was excluded as some subjects did not complete some/all tasks, did not produce correct responses for the required tasks, did not record the start and end time, and/or did provide an unrealistic time. Solutions containing minor syntactic errors or typos are retained as long as they make sense semantically. Furthermore, since actor stability is computed between two GRL consecutive versions, the output of the current task, is considered as the base to the next task. We discard the output of a task

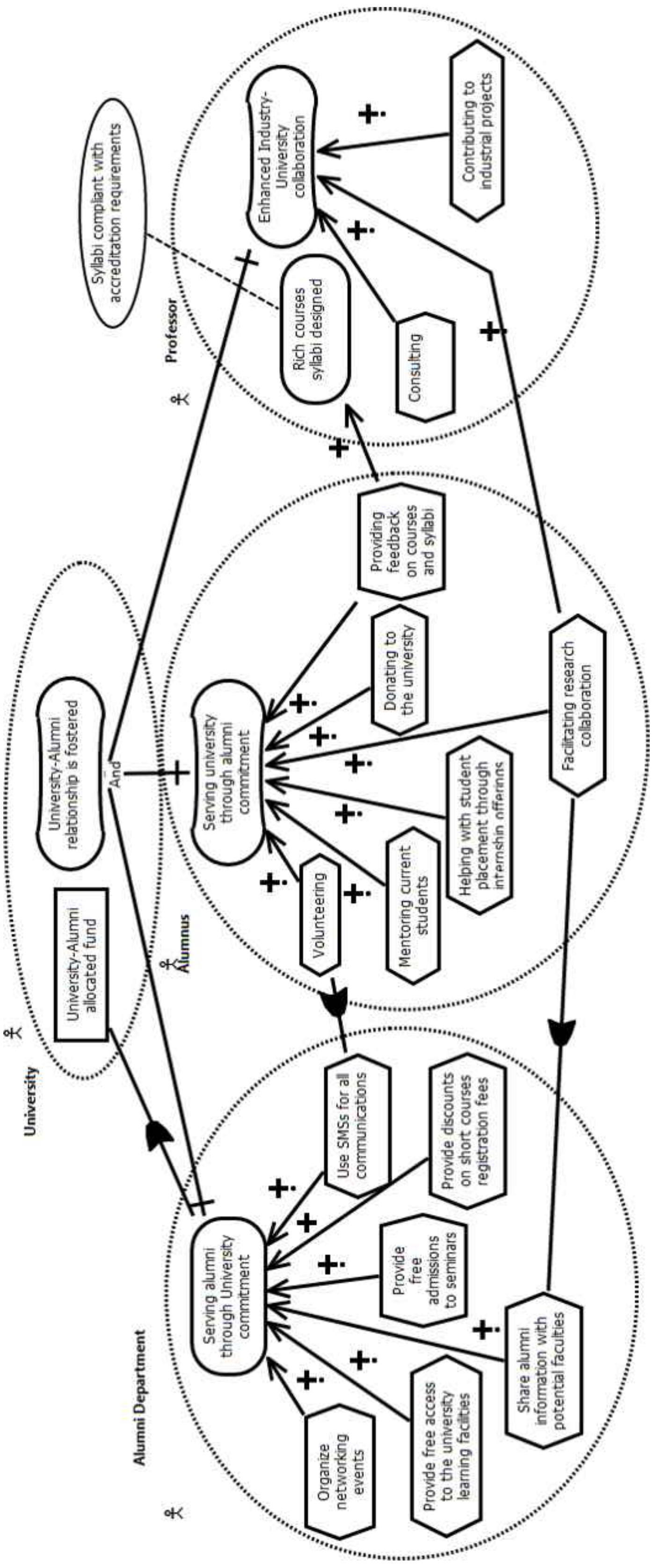

Alumni Department
University
Alumnus
Professor
Serving alumni through University commitment
Organize networking events
Provide free access to the university learning facilities
Share alumni information with potential faculties
Provide free admissions to seminars
Provide discounts on short courses registration fees
Use SMSs for all communications
University-Alumni allocated fund
University-Alumni relationship is fostered
And
Serving university through alumni commitment
Volunteering
Mentoring current students
Helping with student placement through internship offerings
Facilitating research collaboration
Donating to the university
Providing feedback on courses and syllabi
Rich courses syllabi designed
Syllabi compliant with accreditation requirements
Enhanced Industry-University collaboration
Consulting
Contributing to industrial projects

Figure 6: Initial GRL model describing the fostering of the university – alumni relationship

that results in an invalid GRL model. However, if a task output is syntactically valid but represents an incorrect solution then we consider it as a base model for the subsequent task since the tasks are independent.

Figure 7 illustrates an example of the resulting "Alumni Department" actor performed by one of the groups as part of maintenance Task 1 and Table 4 shows its corresponding stability computation. In order to address the first issue, the goal "Serving alumni through University commitment" is converted to a softgoal. To address the second issue, the contribution type between task "Use SMSs for all communications" and softgoal "Serving alumni through University commitment" is changed from "help" to "hurt".

In the original model, the Alumni department has 8 GCUs (i.e. 6 GCUs with tasks and contributions, one GCU composed of a goal and a dependency and one GCU composed of one goal and one AND decomposition link). By converting the goal to a softgoal both GCUs involving the goal are considered as changed (i.e. number of unchanged goal GCUs is zero). Among the 6 GCUs involving tasks, only "use SMS for all communications" task and the *help* contribution (converted to a *hurt*) has changed, i.e. the number of unchanged task GCUs is 5. The number of distinct types of intentional elements is equal to 2 for the Alumni Department (goals and tasks). Hence, the stability of the "Alumni Department" actor is 0.416.

Figure 8 illustrates an example of the resulting "Alumnus" actor performed by one of the groups as part of maintenance Task 3 and Table 5 shows its corresponding stability computation. To address the issue of mentoring a large number of students, the task "mentoring current students" is changed to "Mentoring a maximum of 2 students per year". Another possible change would be to keep the task as is and change the contribution type from *help* to *hurt* (not shown in Figure 8).

In the original model, the Alumnus actor has 11 GCUs (i.e. 8 GCUs with tasks and contributions, 2 GCUs composed of a task and a dependency and one GCU composed of one softgoal and one AND decomposition link). Only one GCU, having a task as intentional element, is changed leaving the 9 task GCUs unchanged. The number of distinct types of intentional elements is equal to 2 for the Alumnus actor (softgoals and tasks). Hence, the stability of the Alumnus actor is 0.95.

The University and Professor actors are fully stable (i.e. stability equal to 1) since they have not been changed as part of the four maintenance tasks, while Alumnus and Alumni Department actors are partially stable.

5.2.4. Experimental analysis

To test the hypothesis, we performed Spearman correlation between the actor stability value and maintainability effort measured by the time spent on each task (in seconds). We considered each task as a different experiment and thus the data was combined to perform the analysis. Results of performing a maintenance task can vary be-

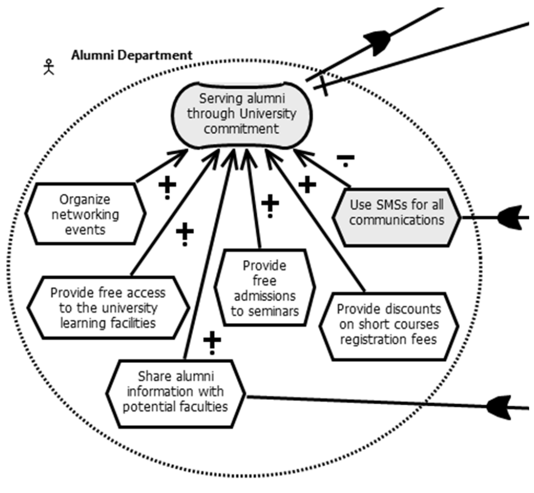

Alumni Department
Serving alumni through University commitment
Organize networking events
Provide free access to the university learning facilities
Provide free admissions to seminars
Use SMSs for all communications
Provide discounts on short courses registration fees
Share alumni information with potential faculties

Figure 7. Example of maintenance Task 1 solution

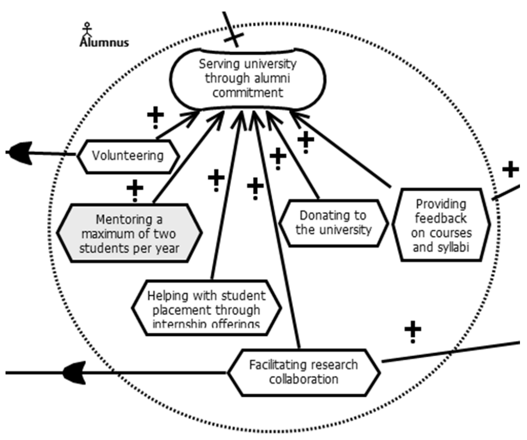

Alumnus
Serving university through alumni commitment
Volunteering
Mentoring a maximum of two students per year
Donating to the university
Providing feedback on courses and syllabi
Helping with student placement through internship offerings
Facilitating research collaboration

Figure 8. Example of maintenance Task 3 solution

Table 4. Maintenance Task 1 stability computation

Actor: Alumni department		
GCU intentional element	Maximum possible change	Number of unchanged GCUs
Goals	2	0
Softgoals	–	–
Tasks	6	5
Resources	–	–
Beliefs	–	–
ExtentOfChange: (0/2) + (5/6) = (5/6)		
Distinct types of intentional elements: 2		
GRL actor stability: (5/6) / 2 = 0.416		
Note: "–" means the intentional element is not present in the Actor.		

Table 5. Maintenance Task 3 stability computation

Actor: Alumnus		
GCU intentional element	Maximum possible change	Number of unchanged GCUs
Goals	–	0
Softgoals	1	1
Tasks	10	9
Resources	–	–
Beliefs	–	–
ExtentOfChange: (1/1) + (9/10) = (19/10)		
Distinct types of intentional elements: 2		
GRL actor stability: (19/10) / 2 = 0.95		
Note: "–" means the intentional element is not present in the Actor.		

tween subjects as their solutions might be different. Therefore, the time recorded to perform the maintenance task can also vary by different subjects. However, the stability measurement of the solution can be equal. The results of the experiment, shown in Figure 9, show that the maintainability effort has a significant strong negative correlation as the correlation value is -0.713 and the P-value is <0.05 [72]. Thus, we reject the null hypothesis and accept the alternative hypothesis, that there is a correlation between actor stability and maintainability measured by time spend on performing the maintenance task.

The results confirm that there is a negative relationship between GRL actor stability and maintenance effort, hence, the less stable the actor, the more maintenance effort it requires. Therefore, actor stability value can be used as an indicator of maintenance effort. Requirements engineers need to give the GRL model special attention when the stability value is low as this will yield to high maintenance cost.

5.2.5. Threats to validity

The proposed metric and its empirical validation are subject to some limitations and threats to validity that we categorize as follows:

Conclusion validity: a possible threat is the small sample size used in the validation. More samples would have provided more confidence in the evaluation. Another possible conclusion threat is that we used a simple case study; this should not affect the validity of the results, as the metric will be measured in the same way regardless the systems' size. However, in our future work, our plans include conducting an experiment using bigger real-world case studies to further support our findings. A third possible conclusion threat is the re-

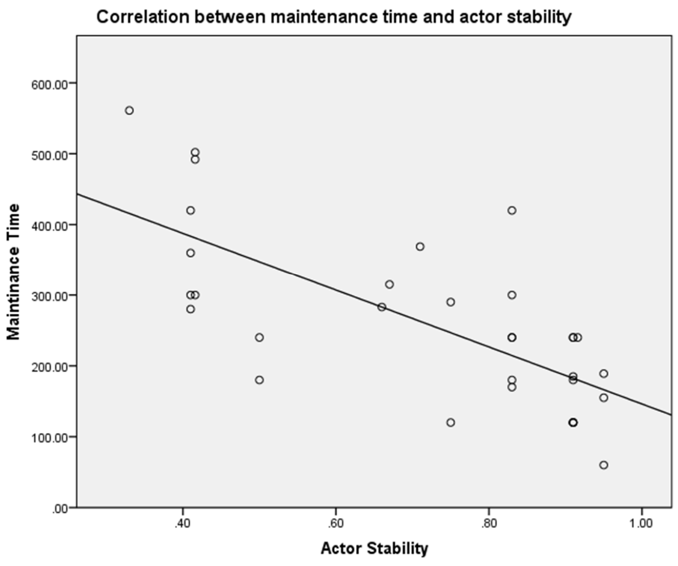


Figure 9. Correlation between maintenance time and actor stability

liability of the time measurement recorded by the subject. Although how to measure the time span was clarified to the subjects, variation in time measurement may have occurred. The last possible conclusion validity threat is that we correlated the actor stability values with maintainability effort measured by the time it takes to perform a task. However, we assumed that this relation exists based on the relationship identified by ISO 25010.

Internal validity: a possible internal threat is that some subjects did not perform the task correctly and thus produced a wrong GRL model that does not satisfy the task requirement. The data of such tasks was excluded from the empirical validation as they produced invalid GRL model. However, to mitigate this threat, we provided all subjects with similar training in GRL language including hands-on using the jUCMNav tool. Another possible internal threat is related to the variables used in the empirical validation; we used the time spent by the subjects in performing the four maintenance tasks and the stability values of the actors as dependent variables. However, the selection of these variables is done based on the existing relationship reported in ISO 25010. ISO 25010 indicates that maintainability is related to modifiability which in turns is related to stability. Thus, we expect maintainability to be correlated with stability. Furthermore, there is a threat that the time for solving latter tasks might be affected by the learning time spent on earlier tasks. However, this practice effect is experienced by all participants as we followed the same sequence of tasks in all experiment tasks with all subjects.

Construct validity: a possible construct validity threat is that we consider all intentional elements to have the same weight regardless of the number of instances each element has; however, this is done intentionally to make sure that we treat all intentional elements equally as some intentional elements might be used more than others in GRL models. In fact, this is the reason why we did not consider a simpler metric that measures the number of unchanged GCUs divided by the total number of GCUs.

External validity: the subjects who performed the experiments are undergraduate software engineering students. The subjects executed the experiment tasks and recorded the time taken to perform each task. This presents an external threat as the experiments were not performed by professionals, however, a recent study by Falessi et al. [73] shows that using students as subjects is acceptable and provides simplification of the actual context. Another possible external threat is related to the tasks used in the empirical evaluation as they are not industry tasks. To mitigate this threat, we planned and designed the maintenance tasks carefully so that they are not trivial and require careful analysis to mimic real maintenance tasks in addition to involving changes on different intentional elements.

6. Discussion

In the following subsections, we discuss the benefits of our proposed approach and how to interpret the metric.

6.1. Metric benefits

In early requirements engineering process, goal models are used to capture interests, intentions and strategies of different stakeholders. They go through many modifications that are necessary to accommodate changing user requirements, evolving business goals and objectives or even induced by changes in implementation technologies. The proposed GRL actor stability metric brings the following benefits:

1. It offers a systematic way to measure the extent of modifications across many versions of a goal model.
2. The computation of the metric is based on easily countable parameters, such as the number of unchanged GCUs, that does not require individual attention or time-consuming processing.
3. It allows for reasoning about which actor is less/more stable. In case, an actor represents a human stakeholder, an instable actor may be an indication that your stakeholders do not understand the problem they are trying to address, as they have changed their minds drastically. Some sort of visioning session with them may be necessary. In addition, it allows for an early assessment of the risk of a major project reset as a result of several new stakeholder input.
4. The proposed metric takes into consideration all GRL constructs. We believe that the computation of the stability metric can be fully automated in this context.
5. It can be generalized to cover other goal-oriented languages, such as i* [5], KAOS [7] and TROPOS [8]. Indeed, our approach is based on the notion of GCU, which is present in i* Strategic Rationale (SR) diagrams. The KAOS approach covers goals of many types but is less concerned with the intentionality of actors. However, our stability metric may be applied at the goal model level to assess the extent of changes between different versions of the model. Similarly, we may tweak the metric to cover TROPOS and i* Strategic Dependency (SD) models. Indeed, the relationships between actors and other constructs in i* SD models and in TROPOS can be considered as a GCU, which is the basic concept in our proposed metric.

6.2. Metric practical implication

The proposed GRL actor stability metric can be used as a proxy of maintenance effort. Our empirical validation of the proposed metric has shown a direct negative relationship between actor stability and maintenance effort. Hence, requirements engineers may have an indicator of the maintenance effort required to maintain the GRL model. A low stability value indicates that the model will require more maintenance effort, therefore, the requirements engineers can make appropriate actions to refactor the current GRL model in order to reduce the expected maintenance effort.

7. Conclusion and future work

Requirements evolution is a main driver for systems evolution. Many metrics have been proposed to understand the sources, frequencies and types of requirements evolution. More specifically, many metrics have been introduced to measure requirements stability at different abstraction and granularity levels. Goal models are used to capture interests, intentions and strategies of different stakeholders in early requirements engineering. In this paper, we presented a novel metric to measure GRL actor stability. The proposed metric provides a quantitative indicator of GRL actor maintainability to have a better estimation of the change cost. We have validated theoretically and empirically our proposed stability metric using a case study.

As a future work, we plan to automate and apply the proposed metric to real-world large-size case studies to assess whether our metric is a good indicator of the stability of GRL actors. We also plan to investigate which type of maintenance effort has the highest impact on stability. In addition, we plan to build prediction models to predict GRL actor stability. Furthermore, we plan to conduct an empirical experiment to study if stability measures converge over time and have a consistent trend. Moreover, we are currently working on proposing a metric suite for goal-oriented languages, which includes model stability. In this paper, we used time to measure maintainability, in future studies, we plan to use other measures such as effort.

Acknowledgment

The authors would like to acknowledge the support provided by King Fahd University of Petroleum & Minerals (KFUPM) for funding this work.

References

[1] J.C. Chen and S.J. Huang, "An empirical analysis of the impact of software development problem factors on software maintainability," *Journal of Systems and Software*, Vol. 82, No. 6, 2009, pp. 981–992.

[2] D. Galorath, "Software total ownership costs: development is only job one," *Software Tech News*, Vol. 11, No. 3, 2008.

[3] J. Li, H. Zhang, L. Zhu, R. Jeffery, Q. Wang, and M. Li, "Preliminary results of a systematic review on requirements evolution," *IET Conference Proceedings*, 2012, pp. 12–21.

[4] ISO/IEC, "25010:2011: Systems and software engineering – systems and software quality requirements and evaluation," 2011.

[5] E.S. Yu, "Towards modelling and reasoning support for early-phase requirements engineering," in *International Symposium on Requirements Engineering.* IEEE, 1997, pp. 226–235.

[6] L. Chung and J. Leite, *On Non-Functional Requirements in Software Engineering.* Springer, 2009, pp. 363–379.

[7] A. van Lamsweerde, "Requirements engineering: from craft to discipline," in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering.* ACM, 2008, pp. 238–249.

[8] P. Giorgini, J. Mylopoulos, and R. Sebastiani, "Goal-oriented requirements analysis and reasoning in the tropos methodology," *Engineering Applications of Artificial Intelligence*, Vol. 18, No. 2, 2005, pp. 159–171.

[9] ITU-T, "Recommendation Z.151 (10/18), User Requirements Notation (URN) language definition, Geneva, Switzerland," Geneva, Switzerland, 2018. [Online]. http://www.itu.int/rec/T-REC-Z.151/en

[10] S. Overbeek, U. Frank, and C. Köhling, "A language for multi-perspective goal modelling: Challenges, requirements and solutions," *Computer Standards & Interfaces*, Vol. 38, 2015, pp. 1–16. [Online]. http://www.sciencedirect.com/science/article/pii/S0920548914000798

[11] A. Dias, V. Amaral, and J. Araujo, "Towards a domain specific language for a goal-oriented approach based on KAOS," in *Third International Conference on Research Challenges in Information Science.* IEEE, 2009, pp. 409–420.

[12] D. Quartel, W. Engelsman, H. Jonkers, and M. van Sinderen, "A goal-oriented requirements modelling language for enterprise architecture," in *International Enterprise Distributed Object Computing Conference.* IEEE, 2009, pp. 3–13.

[13] X. Franch and N. Maiden, "Modelling component dependencies to inform their selection," *COTS-Based Software Systems*, Vol. 2580 of LNCS, 2003, pp. 81–91.

[14] X. Franch, G. Grau, and C. Quer, "A framework for the definition of metrics for actor-dependency models," in *12th International Requirements Engineering Conference.* IEEE, 2004, pp. 348–349.

[15] X. Franch, "On the quantitative analysis of agent-oriented models," *Advanced Information Systems Engineering*, Vol. 4001 of LNCS, 2006, pp. 495–509.

[16] G. Grau, X. Franch, and N. Maiden, "Prim: An i*-based process reengineering method for information systems specification," *Information and Software Technology*, Vol. 50, No. 1-2, 2008, pp. 76–100.

[17] H. Kaiya, H. Horai, and M. Saeki, "Agora: attributed goal-oriented requirements analysis method," *Joint International Conference on Requirements Engineering*, 2002, pp. 13–22.

[18] G. Grau and X. Franch, "A goal-oriented approach for the generation and evaluation of alternative architectures," in *European Confer-*

ence on Software Architecture. Springer, 2007, pp. 139–155.

[19] A. van Lamsweerde, "Goal-oriented requirements engineering: A guided tour," in *Proceedings fifth International Symposium on Requirements Engineering.* IEEE, 2001, pp. 249–262.

[20] A.I. Antón, W.M. McCracken, and C. Potts, "Goal decomposition and scenario analysis in business process reengineering," in *International Conference on Advanced Information Systems Engineering.* Springer, 1994, pp. 94–104.

[21] C.M. Nguyen, R. Sebastiani, P. Giorgini, and J. Mylopoulos, "Multi-objective reasoning with constrained goal models," *Requirements Engineering*, 2016, pp. 1–37.

[22] J. Horkoff, F.B. Aydemir, E. Cardoso, T. Li, A. Maté, E. Paja, M. Salnitri, J. Mylopoulos, and P. Giorgini, "Goal-oriented requirements engineering: A systematic literature map," in *24th International Requirements Engineering Conference (RE).* IEEE, 2016, pp. 106–115.

[23] T. Ambreen, N. Ikram, M. Usman, and M. Niazi, "Empirical research in requirements engineering: trends and opportunities," *Requirements Engineering*, 2016, pp. 1–33.

[24] L. López, F.B. Aydemir, F. Dalpiaz, and J. Horkoff, "An empirical evaluation roadmap for iStar 2.0," in *Proceedings of the Ninth International i* Workshop (istar'16)*, Vol. 1674, 2016, pp. 55–60.

[25] M.A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, and P. González, "Comparing goal-oriented approaches to model requirements for CSCW," in *International Conference on Evaluation of Novel Approaches to Software Engineering.* Springer, 2011, pp. 169–184.

[26] J.P. Carvallo and X. Franch, "On the use of i* for architecting hybrid systems: A method and an evaluation report," in *IFIP Working Conference on The Practice of Enterprise Modeling.* Springer, 2009, pp. 38–53.

[27] G. Elahi, E. Yu, and M.C. Annosi, "Modeling knowledge transfer in a software maintenance organization – an experience report and critical analysis," in *IFIP Working Conference on The Practice of Enterprise Modeling.* Springer, 2008, pp. 15–29.

[28] K. Hoesch-Klohe, *A Framework to support the Maintenance of Formal Goal Models*, Ph.D. dissertation, University of Wollongong, 2013. [Online]. http://ro.uow.edu.au/theses/4214

[29] N.A. Ernst, J. Mylopoulos, and Y. Wang, *Requirements Evolution and What (Research) to Do about It.* Berlin, Heidelberg: Springer, 2009, pp. 186–214.

[30] A.K. Ghose, "Formal tools for managing inconsistency and change in RE," in *Proceedings of the 10th International Workshop on Software Specification and Design.* IEEE Computer Society, 2000, p. 171.

[31] N.A. Ernst, A. Borgida, J. Mylopoulos, and I.J. Jureta, *Agile Requirements Evolution via Paraconsistent Reasoning.* Berlin, Heidelberg: Springer, 2012, pp. 382–397.

[32] A.M. Grubb and M. Chechik, "Looking into the crystal ball: requirements evolution over time," in *24th International Requirements Engineering Conference (RE).* IEEE, 2016, pp. 86–95.

[33] Aprajita and G. Mussbacher, "TimedGRL: Specifying goal models over time," in *24th International Requirements Engineering Conference Workshops (REW)*, 2016, pp. 125–134.

[34] M.O. Elish and D. Rine, "Investigation of metrics for object-oriented design logical stability," in *Seventh European Conference onSoftware Maintenance and Reengineering.* IEEE, 2003, pp. 193–200.

[35] N.L. Soong, "A program stability measure," in *Proceedings of the 1977 Annual Conference.* ACM, 1977, pp. 163–173.

[36] M. Fayad, "Accomplishing software stability," *Communications of the ACM*, Vol. 45, No. 1, 2002, pp. 111–115.

[37] S.S. Yau and J.S. Collofello, "Some stability measures for software maintenance," *IEEE Transactions on Software Engineering*, Vol. SE-6, No. 6, 1980, pp. 545–552.

[38] W. Li, L. Etzkorn, C. Davis, and J. Talburt, "An empirical study of object-oriented system evolution," *Information and Software Technology*, Vol. 42, No. 6, 2000, pp. 373–381.

[39] M. Alshayeb and W. Li, "An empirical study of system design instability metric and design evolution in an agile software process," *Journal of Systems and Software*, Vol. 74, No. 3, 2005, pp. 269–274.

[40] A. AbuHassan and M. Alshayeb, "A metrics suite for UML model stability," *Software Systems Modeling*, Vol. 18, No. 1, 2019, pp. 557–583.

[41] Y. Hassan, *Measuring software architectural stability using retrospective analysis*, Ph.D. dissertation, King Fahd University of Petroleum & Minerals, 2007.

[42] J. Bansiya, "Evaluating framework architecture structural stability," *ACM Computing Surveys*, Vol. 32, No. 1, 2000.

[43] M. Mattsson and J. Bosch, "Characterizing stability in evolving frameworks," in *Technology of Object-Oriented Languages and Systems*, 1999, pp. 118–130.

[44] S.A. Tonu, A. Ashkan, and L. Tahvildari, "Evaluating architectural stability using a metric-based approach," in *Conference on Software Maintenance and Reengineering (CSMR'06)*. IEEE, 2006, pp. 261–270.

[45] D. Grosser, H.A. Sahraoui, and P. Valtchev, "An analogy-based approach for predicting design stability of Java classes," in *5th International Workshop on Enterprise Networking and Computing in Healthcare Industry*. IEEE, 2004, pp. 252–262.

[46] D. Rapu, S. Ducasse, T. Gîrba, and R. Marinescu, "Using history information to improve design flaws detection," in *Eighth European Conference on Software Maintenance and Reengineering*. IEEE, 2004, pp. 223–232.

[47] M. Alshayeb, M. Naji, M. Elish, and J. Al-Ghamdi, "Towards measuring object-oriented class stability," *Software, IET*, Vol. 5, No. 4, 2011, pp. 415–424.

[48] R.C. Martin, "Large scale stability," *C++ Report*, Vol. 9, No. 2, 1997, pp. 54–60.

[49] D. Grosser, H.A. Sahraoui, and P. Valtchev, "Predicting software stability using case-based reasoning," in *Proceedings 17th International Conference on Automated Software Engineering,*. IEEE, 2002, pp. 295–298.

[50] V. Basili, G. Caldiera, and H.D. Rombach, "The goal question metric approach," *Encyclopedia of Software Engineering*, 1994.

[51] X. Franch, *A method for the definition of metrics over i* models*. Berlin Heidelberg: Springer, 2009, Vol. 5565, pp. 201–215.

[52] P. Espada, M. Goulão, and J. Araújo, "A framework to evaluate complexity and completeness of KAOS goal models," in *International Conference on Advanced Information Systems Engineering*. Springer, 2013, pp. 562–577.

[53] C. Gralha, J. Araújo, and M. Goulão, "Metrics for measuring complexity and completeness for social goal models," *Information Systems*, Vol. 53, 2015, pp. 346–362.

[54] J. Hassine and M. Alshayeb, "Measurement of actor external dependencies in GRL models," in *Proceedings of the Seventh International i* Workshop co-located with the 26th International Conference on Advanced Information Systems Engineering*, 2014. [Online]. http://ceur-ws.org/Vol-1157/paper22.pdf

[55] *jUCMNav – Eclipse plugin for Use Case Maps*, University of Ottawa, Canada, 2016. [Online]. http://softwareengineering.ca/jucmnav Last accessed Jan. 2019.

[56] W. Lam and V. Shankararaman, "Requirements change: A dissection of management issues," in *25th EUROMICRO Conference. Informatics: Theory and Practice for the New Millennium*, Vol. 2. IEEE, 1999, pp. 244–251.

[57] S. Anderson and M. Felici, "Quantitative aspects of requirements evolution," in *Proceedings 26th Annual International Computer Software and Applications*. IEEE, 2002, pp. 27–32.

[58] G. Stark, A. Skillicorn, and R. Smeele, "A micro and macro based examination of the effects of requirements changes on aerospace software maintenance," in *Aerospace Conference*, Vol. 4. IEEE, 1998, pp. 165–172.

[59] M. Mattsson and J. Bosch, "Stability assessment of evolving industrial object-oriented frameworks," *Journal of Software Maintenance: Research and Practice*, Vol. 12, No. 2, 2000, pp. 79–101.

[60] L.C. Briand, J.W. Daly, and J. Wüst, "A unified framework for cohesion measurement in object-oriented systems," *Empirical Software Engineering*, Vol. 3, No. 1, 1998, pp. 65–117.

[61] L. Briand, S. Morasca, and V. Basili, "Property-based software engineering measurement," *IEEE Transactions on Software Engineering*, Vol. 22, No. 1, 1996, pp. 68–86.

[62] E.J. Weyuker, "Evaluating software complexity measures," *IEEE Transactions on Software Engineering*, Vol. 14, No. 9, 1988, pp. 1357–1365.

[63] G. Poels and G. Dedene, "Distance-based software measurement: necessary and sufficient properties for software measures," *Information & Software Technology*, Vol. 42, No. 1, 2000, pp. 35–46.

[64] B. Kitchenham, S.L. Pfleeger, and N. Fenton, "Towards a framework for software measurement validation," *IEEE Transactions on Software Engineering*, Vol. 21, No. 12, 1995, pp. 929–944.

[65] C. Wohlin, P. Runeson, M. Host, M. Ohlsson, B. Regnell, and A. Wesslen, *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers, 2000.

[66] A.B. Binkley and S.R. Schach, "Validation of the coupling dependency metric as a predictor of run-time failures and maintenance measures," in *Proceedings of the 20th International Conference on Software ERngineering*. IEEE, 1998, pp. 452–455.

[67] D. Darcy, C. Kemerer, S. Slaughter, and J. Tomayko, "The structural complexity of software: An experimental test," *IEEE Transactions on Software Engineering*, Vol. 31, No. 11, 2005, pp. 982–995.

[68] W. Li and S. Henry, "Object-oriented metrics that predict maintainability," *Journal of Systems*

and Software, Vol. 23, 1993, pp. 111–122.

[69] M. Alshayeb and W. Li, "An empirical validation of object-oriented metrics in two iterative processes," *IEEE Transactions on Software Engineering*, Vol. 29, No. 11, 2003, pp. 1043–1049.

[70] N. Fenton and J. Bieman, *Software Metrics: A Rigorous and Practical Approach.* London: CRC Press, 2014.

[71] J. Hassine and D. Amyot, "A questionnaire-based survey methodology for systematically validating goal-oriented models," *Requirements Engineering*, Vol. 21, No. 2, 2016, pp. 285–308.

[72] J. Evans, *Straightforward Statistics for the Behavioral Sciences.* Brooks/Cole Publishing, 1996.

[73] D. Falessi, N. Juristo, C. Wohlin, B. Turhan, J. Münch, A. Jedlitschka, and M. Oivo, "Empirical software engineering experts on the use of students and professionals in experiments," *Empirical Software Engineering*, Vol. 23, No. 1, 2018, pp. 452–489.

e-Informatica Software Engineering Journal, Volume 13, Issue 1, 2019, pages: 227–259, DOI 10.5277/e-Inf190107

Software Change Prediction: A Systematic Review and Future Guidelines

Ruchika Malhotra*, Megha Khanna**
*Department of Computer Science & Engineering, Delhi Technological University
**Sri Guru Gobind Singh College of Commerce, University of Delhi
ruchikamalhotra@dtu.ac.in, meghakhanna86@gmail.com

Abstract

Background: The importance of Software Change Prediction (SCP) has been emphasized by several studies. Numerous prediction models in literature claim to effectively predict change-prone classes in software products. These models help software managers in optimizing resource usage and in developing good quality, easily maintainable products.
Aim: There is an urgent need to compare and assess these numerous SCP models in order to evaluate their effectiveness. Moreover, one also needs to assess the advancements and pitfalls in the domain of SCP to guide researchers and practitioners.
Method: In order to fulfill the above stated aims, we conduct an extensive literature review of 38 primary SCP studies from January 2000 to June 2019.
Results: The review analyzes the different set of predictors, experimental settings, data analysis techniques, statistical tests and the threats involved in the studies, which develop SCP models.
Conclusion: Besides, the review also provides future guidelines to researchers in the SCP domain, some of which include exploring methods for dealing with imbalanced training data, evaluation of search-based algorithms and ensemble of algorithms for SCP amongst others.

Keywords: change-proneness, machine learning, software quality, systematic review

1. Introduction

The importance of planning and implementing change in a software is accepted universally. It is crucial for a software to reform in order to remove existing defects, to upgrade itself with the changing user requirements and technological progressions or to improve the current performance and structure [1–3]. In case a software product is unable to do so, it rapidly becomes obsolete and extinct. Thus, change management of a software product is a vital activity, which needs to be properly enforced.

Prediction of change-prone parts of a software product is an effective mechanism for software change management. Change-proneness is defined as the likelihood that a class would change across different versions of a software product [1, 4]. Since it indicates whether a specific class would require change in the forthcoming release of the software, it is generally represented by a binary variable indicating “yes” (change-prone class) or “no” (not change-prone class). Knowledge of change-prone classes aids software managers in effectively planning critical software resources such as cost, time and human effort. Sufficient allocation of these resources to change-prone classes ensures that they are carefully designed and rigorously verified [1–3]. Such activities would result in a good quality, easily maintainable and cost-effective software products.

Various studies in literature have successfully developed software quality models to predict change-prone classes of a software. These studies have explored a variety of predictor variables,

Submitted: 3 July 2019; Revised: 14 October 2019; Accepted: 14 October 2019; Available online: 12 November 2019

numerous classification algorithms and extensive software datasets for empirical validation. At such a stage, it is imperative for researchers to analyze the current state of literature and compare the capabilities of existing SCP models in literature. Such a step is important in order to summarize the existing trends in the domain and simultaneously analyze the shortcomings and future directions in the area. Thus, we conduct a systematic literature review of SCP studies from the period of January 2000 to June 2019. The review is conducted according to the guidelines specified by Kitchenham et al. [5].

Though, software change-proneness prediction studies have been earlier assessed by previous review studies [6, 7], they have not been effectively scrutinized for trends specific to the software change domain. These studies have either been explored as an application of Search-Based Algorithms (SBA) to predictive modeling [6] or for only assessing the threats specific to SBA [7]. A previous study by Godara and Singh [8] assessed change-proneness prediction in Object-Oriented (OO) software. However, they only presented a survey of the studies without critically analyzing their various parameters. Also, though a previous attempt by the authors [9] evaluated SCP studies, the analysis was limited with no emphasis on the predictive capabilities of various data analysis algorithms. To the best of our knowledge, no study in the literature has comprehensively evaluated and summarized the experimental settings, predictor metrics, datasets, capabilities of data analysis algorithms, statistical tests and threats with respect to SCP studies.

On the other hand, there are various review studies which analyze defect-proneness prediction (a closely related area to change-proneness) literature. A study by Catal and Diri [10] reviewed 74 defect prediction studies to assess the metrics, data analysis algorithms and datasets used in defect prediction literature. They concluded that method-level metrics, machine learning algorithms and public datasets are the most dominant in the area. Hall et al. [11] reviewed 36 primary studies to study the context, predictors and data analysis algorithms used for defect prediction. They concluded that combination of predictor variables yields better defect prediction models and feature selection enhances the performance of the developed models. Radjenovic et al. [12] reviewed 106 primary studies and classified the metrics used in defect prediction literature as Object-Oriented (OO) metrics, traditional source code metrics and process metrics. A review study by Wahono [13] assessed 71 defect prediction studies between the period January 2000 to December 2013. The review assessed the trends and frameworks in defect prediction literature apart from datasets and data analysis algorithms. As pointed out in the review, the frameworks developed by certain defect prediction studies do not address the issue of class imbalance and noisy data. Hosseini et al. [14] reviewed 30 primary studies to assess the state of the art in cross project defect prediction. They concluded that cross-project defect prediction still requires extensive research before it yields reliable results. Certain other reviews on defect-proneness prediction includes the one conducted by Malhotra [15], Singh et al. [16] and Catal [17]. Though these reviews yield a significant contribution to defect prediction literature, there a huge gap in change prediction literature in terms of an effective and extensive review.

We investigate the following Research Questions (RQs) in the review:

- RQ1: Which predictors are useful for developing SCP models?
- RQ2: What have been the various experimental settings used in primary studies while developing SCP models?
  - RQ2.1: What are the various feature selection or dimensionality reduction techniques?
  - RQ2.2: What are the characteristics of datasets used?
  - RQ2.3: What are the various validation methods used?
  - RQ2.4: Which performance measures have been used?
- RQ3: What are the various categories of data analysis algorithms used for developing SCP models?
  - RQ3.1: Which is the most popular category of data analysis algorithm used?

  - RQ3.2: Which Machine Learning (ML) algorithms have been evaluated?
- RQ4: What is the predictive performance of ML algorithms for developing SCP models? How does the predictive capability of ML algorithms compare amongst themselves?
- RQ5: What are the various statistical tests used for validating the results of SCP models?
- RQ6: What threats to validity exist while developing SCP models?
  - RQ6.1: What are the various categories of threats which exist while developing SCP models?
  - RQ6.2: What are the steps required to mitigate the threats identified in RQ6.1?

The aim of the review is to summarize the empirical evidence reported in literature with respect to SCP. It would also help in identification of research gaps and will provide future possible guidelines to researchers and practitioners. The organization of the study includes the importance of SCP (Section 2), review procedure and the various review stages (Section 3), the review protocol (Section 4), the answers to the investigated RQ's (Section 5), the threats (Section 6) and finally the conclusions and future guidelines (Section 7).

2. Importance of SCP

There are several diverse reasons for change-proneness of a specific code segment. Some real-world examples of why a specific code segment could be prone to changes are provided below:

- A code segment may have bad structure or rigid design which is difficult to extend [18].
- A code segment might contain errors which have escaped the testing phase and now requires maintenance [19–21].
- Business requirements of an organization could change necessitating a change in source code segment [19, 20].

Therefore, SCP is critical in order to identify such change-prone code segments in the early stages of software lifecycle so that developers allocate proper manpower, cost and time to modify them [18, 21–23]. Such a step is crucial in order to keep the software operational, and ensure customer satisfaction. Even in the era of agile development, SCP is an approach to continuously monitor change-proneness so that effective product quality is maintained. Neglect of change-prone code fragments could result in poor software quality, extensive costs [20] as the cost to correct errors increases manifold as they propagate to later stages of the software product coupled with delayed delivery schedules.

Before we state the review procedure, it is important to ascertain as to how a specific segment (class/file/module) of source code is adjudged as change-prone or not change-prone in primary studies. Majority of primary studies consider a specific segment as change-prone if there is at least one insertion, removal or modification of Source Lines of Code (SLOC) in the specific code segment from current software release to its next [24–26]. We term this definition as "SLOC-based" method. In certain other primary studies, a specific code segment is considered change-prone if the number of changes it underwent from one release to the next is greater than the median value (median of number of changes in all the source code segments in the software) [19, 27, 28]. We term the definition as "median-based" method. However, few primary studies used boxplot-based partition method [18], class stability [29] or other methods to define the dichotomous change-proneness dependent variable.

As discussed above, primary studies have used various methods to define change-proneness. However, it is difficult to adjudge any one of them as best as each method has its own merits and demerits. The "SLOC-based method" may designate a segment as change-prone even if very few (even one) SLOC has been changed. This ensures that no change is missed as it is possible that critical changes to the code are performed by changing very few SLOC. It may be noted that such code segments may be candidates for corrective maintenance but not very much for preventive maintenance as only few SLOC has been modified and not much structure would have been altered. However, this method has a downside, even trivial changes to the code would also make the code segment counted as change-prone. On the other hand, the "median-based" method ignores the code segments with very few changes. It focuses on identifying classes which requires preventive maintenance. The goal of "median-based" method is to find which classes will change more than others, not which classes will change in an absolute manner. But as discussed earlier the "median-based" method may ignore critical changes if they are performed using very few lines.

3. Review procedure

According to the guidelines advocated by Kitchenham et al. [5], a review is conducted in three fundamental stages. These stages are reportedly planning, conducting and reporting. The foremost step of the planning stage is to evaluate the necessity of the review. As already discussed, this review is important so as to evaluate, assess and summarize the empirical evidence with respect to prediction of change-prone classes in software products. It intends to provide an overview of existing literature in the domain and would scrutinize possible future directions. Once the need of the review is assessed, the planning stage involves formation of RQs. Thereafter, a review protocol is formulated. The protocol includes a detailed search strategy. The search strategy consists of the list of possible search databases one intends to scrutinize, the search string and the criteria for inclusion or omittance of the extracted studies. Apart from the search strategy, the protocol also includes the criteria for assessing the quality of the candidate studies, the procedure for collecting the relevant data from the primary studies and synthesis of the collected data. The second stage involves the actual execution of the review protocol. In this stage, all the primary studies are extracted, scrutinized and the relevant data is obtained. The final stage of the review reports the results of the investigated RQs. The RQs are answered on the basis of the data collected from the primary studies of the review.

4. Protocol for conducting the review

The following sections describe the review protocol followed which lists the search strategy used, the criteria used for selecting or omitting the extracted studies and the criteria for evaluating the quality of the collected candidate studies.

4.1. Search strategy

The search terms were designed by dividing the explored RQs into logical units. Moreover, terms were identified from paper titles, keywords and abstracts. Thereafter, all equivalent terms and synonyms were compiled using Boolean OR (||), while distinguishable search terms were aggregated using Boolean AND (&). As indicated earlier, the search period was January 2000–June 2019. The designed search-string is:

("software product" || "open source project" || "software application" || "software system" || "software quality" || "source code") & ("change" || "evolution" || "maintenance") & ("prediction" || "proneness" || "classification" || "classifier" || "empirical") & ("machine learning" || "statistical" || "search-based" || "evolutionary" || "data analysis")

The search was conducted in SCOPUS, ACM digital library, Wiley online library, IEEExplore and SpringerLink, as these are well-known search databases. We also searched the reference lists of the studies and found seven studies. In all, we identified and extracted 67 relevant studies, which were further subjected to the criteria indicated in Section 4.2.

4.2. Inclusion and omittance criteria

We use the following inclusion and omittance criteria for selecting or rejecting a study based on the RQs, after which we get 41 candidate studies.

– *Inclusion Criteria*: All studies which predict the dichotomous change-proneness attribute of a class/module or determine class stability with the aid of software metrics were included. We also include studies which reported and compared various data analysis algorithms amongst themselves for developing SCP models.

– *Omittance Criteria*: Extracted studies which predict other dependent variables such as maintenance effort, maintainability, change-count, fault-proneness, amount of changes, etc. were excluded. A related concept to change-proneness is code churn. It is defined as the volume of SLOC that is changed (inserted, modified or removed) between two versions of a software and represents the extent of change [30]. It is a continuous attribute and encapsulates the maintenance effort required by the class while it undergoes changes (bug correction, enhancements or refactoring). The current review limits itself to binary change-proneness attribute and does not include studies which assess code churn. Also, studies which predict ordinal dependent variables for change-proneness such as low, medium, high, etc. were not included as a part of the review.
  We also omitted survey or review papers, PhD dissertations, short or poster papers and studies with limited or non-existent empirical analysis. A conference paper which has been published as a journal article was also omitted and only the corresponding journal article was included. Though change-proneness attribute has been explored in design pattern literature [31] and technical debt literature [32], we exclude such studies. Therefore, studies which used only design patterns or code smells for determining change-prone nature of a class /module were removed.

4.3. Quality criteria

We assess the importance of each candidate study in answering the investigated RQs. The 41 candidate studies were evaluated according to the quality criteria illustrated in Table 1. Each candidate study was given a Quality Score (QS) by combining the scores of a specific study on the basis of the 10 quality questions stated in Table 1. For each question, a candidate study can be either given a score of 0 (No), 0.5 (Partly) or 1 (Yes). Table 1 states the number of candidate studies which were allocated different scores (Yes, Partly or No). All the studies whose QS was less than 5 (50% of the total quality score) were rejected. We rejected three studies [33–35]. After quality analysis, we selected 38 studies, which we term as the primary studies of the review. The data needed to answer the RQs was extracted only from the primary studies.

Table 2 states the Primary Studies (PS) with a specific study number (S.No.) and its QS. The most popularly cited studies were PS10 and PS13.

4.4. Data extraction

The primary studies were classified according to publication year, publication venue, predictors, datasets, data analysis algorithms, performance measures, validation methods, statistical tests and threats to validity.Table A1 (Appendix A)

Table 1. Quality assessment questions

Quality questions	Yes	Partly	No
Are the objectives of the research/research questions clear and concise?	41	0	0
Are the predictor variables clearly defined and described?	27	12	2
Are the number and magnitude of datasets analyzed suitable?	30	10	1
Are the predictors effectively chosen using feature selection/dimensionality reduction techniques?	20	4	17
Are the data analysis techniques clearly defined and described?	25	9	7
Is there any comparative analysis amongst various models/techniques?	34	1	6
Are the performance measures clearly specified?	33	7	1
Did the study perform statistical hypothesis testing?	25	1	15
Does the study use appropriate validation methods?	32	1	8
Is there a description of threats to validity of research?	19	3	19

Table 2. Primary studies with quality score

S.No.	Study	QS	S.No.	Study	QS
PS1	Liu and Khoshgoftaar 2001 [36]	6.5	PS20	Elish et al. 2017 [37]	8
PS2	Khoshgoftaar et al. 2003 [38]	6.5	PS21	Kumar et al. 2017a [39]	8
PS3	Tsantalis et al. 2005 [40]	8	PS22	Kumar et al. 2017b [41]	9
PS4	Sharafat and Tahvildari 2008 [42]	5.5	PS23	Kumar et al. 2017c [43]	7
PS5	Azar 2010 [44]	6.5	PS24	Malhotra and Jangra 2017 [45]	9
PS6	Han et al. 2010 [46]	6	PS25	Malhotra and Khanna 2017a [26]	9.5
PS7	Azar and Vybihal 2011 [29]	7.5	PS26	Malhotra and Khanna 2017b [47]	9.5
PS8	Eski and Buzluca 2011 [48]	5	PS27	Yan et al. 2017 [49]	9.5
PS9	Lu et al. 2011 [24]	7	PS28	Agrawal and Singh 2018 [50]	8
PS10	Romano and Pinzger 2011 [27]	8	PS29	Catolino et al. 2018 [19]	9.5
PS11	Giger et al. 2012 [28]	8	PS30	Ge et al. 2018 [23]	6.5
PS12	Elish et al. 2013 [25]	9.5	PS31	Liu et al. 2018 [51]	7
PS13	Malhotra and Khanna 2013 [52]	9	PS32	Kaur and Mishra 2018 [53]	8
PS14	Malhotra and Bansal 2014 [54]	6	PS33	Malhotra and Khanna 2018a [55]	9.5
PS15	Malhotra and Khanna 2014 [56]	9	PS34	Malhotra and Khanna 2018b [57]	9.5
PS16	Marinescu 2014 [58]	7	PS35	Zhu et al. 2018 [18]	9.5
PS17	Elish et al. 2015 [59]	6	PS36	Catolino and Ferrucci 2019[19]	9.5
PS18	Malhotra and Khanna 2015 [60]	8.5	PS37	Kumar et al. 2019 [22]	8
PS19	Bansal 2017 [61]	9.5	PS38	Malhotra and Khanna 2019 [21]	9.5

states the key parameters of primary studies after the data extraction step.

5. Review results

The current section states the review results and the discussions corresponding to the obtained results. We categorized the primary studies according to their publication venue and found that 37% of the 38 primary studies were conference publications, 59% of the studies were journal publications and one study each was published as a technical report and a chapter. The most popular journals were "Information and Software Technology" (11% studies) and "Journal of Software: Evolution and Process" (8% studies). "International Conference on Advances in Computing, Communication and Informatics" and "Innovations in Software Engineering Conference" were found to be the most popular conference venues with 5% of publications each. Figure 1 depicts a distribution of all the primary studies according to "publication year". According to the figure, the highest number of SCP studies were published in 2017.

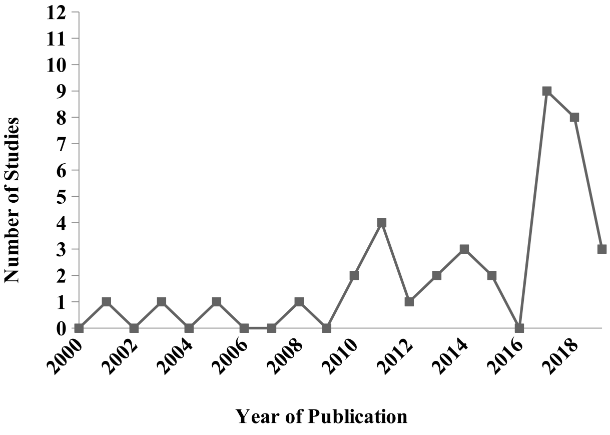


Figure 1. Publication year of primary studies

5.1. Predictors used for SCP (RQ1)

This RQ determines the various predictors which have been used for developing SCP models. An analysis of primary studies reveals that both product as well as process metrics have been used as predictors for SCP. Table 3 lists the various metrics used in primary studies for developing SCP models.

According to Table 3, product metrics especially structural metrics extracted from source code design have been widely used in SCP

Table 3. Predictors used by SCP studies

Metric category	Brief description	Study numbers	Category
Structural metrics	These metrics are generally source code design metrics, which depict the structural attributes of a class such as its inheritance, cohesiveness, size, etc. Many such metric suites have been proposed in literature such as Chidamber and Kemerer (CK) metrics suite [62], Quality Models for Object Oriented Design metrics suite [63], Lorenz and Kidd metrics suite [64], Li and Henry metrics suite [65] and many others.	PS1–PS38	Product
Network metrics	These metrics are extracted from the dependency graph of the software and identifies files which are "more central" and are more likely to change, e.g. Degree centrality, Closeness centrality, Reachability, etc.	PS11, PS35	Product
Evolution based metrics	These metrics characterize evolution history of a class, i.e. release by release history of how a class has evolved in previous versions, e.g. Birth of a Class, Frequency of changes, Change density, etc.	PS12, PS20, PS29, PS33, PS36	Process
Word vector metrics	These metrics quantify the terms used in the source code files and their names by using bag of words approach.	PS35	Product
Developer related metrics	These metrics quantify various developer related factors such as entropy of changes introduced by a developer in a given time period, number of developers employed on a specific software segment in a specific time, structural and semantic scattering of developers in a specific time period, etc., e.g. entropy of changes applied by developers in a given time period, structural scattering of developers that work on a particular class in a given time period, etc.	PS29, PS36	Process
Combination of structural and evolution based metrics	This metric suite combines structural and evolution-based metrics as they quantify two different attributes (software design and evolution history) of a class.	PS12, PS20, PS33	Product and Process
Others	Metrics such as instability and maintainability index used by PS32, probability of change based on inheritance, reference and dependency used by PS3.	PS32, PS3	Product

literature. It was found that all the studies evaluated this category of metrics. Even primary studies which proposed other possible category of metrics (evolution-based, network, developer-related, etc.) assessed and compared their proposed predictors with structural metrics as they are well established and successfully used by numerous studies. We found that the CK metrics suite was the most commonly used structural metrics suite in primary studies, which characterized various OO attributes. A similar observation was stated by Radjenovic et al. [12] while analyzing defect prediction studies. The CK metrics suite consists of Weighted Methods of a Class (WMC), Lack of Cohesion amongst Methods (LCOM), Coupling Between Objects (CBO), Response for a Class (RFC), Depth of Inheritance Tree (DIT) and Number of Children (NOC) metrics. Apart from the CK metric suite, the SLOC metric (a measure of class size) has also been frequently used in primary studies. Other product metrics used were network metrics, word vector metrics and the "others" category.

Only 16% of primary studies used process metrics. While PS12, PS20, PS29, PS33 and PS36 used evolution-based metrics which characterize the evolution history of a class, PS29 and PS36 used metrics which depict the development process complexity by quantifying developer related factors. It may be noted that PS12, PS20 and PS33 advocated the combination of both process as well as product metrics for determining change-prone nature of a class.

We also analyzed the granularity level over which these metrics were collected. Five studies (PS1, PS2, PS11, PS32 and PS35) collected file-level metrics, one study each collected structural metrics at interface level (PS10) and method level (PS4). However, all other studies analyzed class-level metrics. It may also be noted that certain studies (PS5, PS7, PS9, PS22, PS23, PS24), analyzed a large number of OO metrics with respect to different dimensions (cohesion, coupling, size and inheritance) in order to obtain generalized results.

5.2. Experimental settings for SCP (RQ2)

This RQ explores the various experimental settings, i.e. the feature selection or dimensionality reduction methods, the characteristics of datasets used for empirical validation, the validation methods and the performance measures used by SCP studies.

5.2.1. Feature selection and dimensionality reduction techniques (RQ2.1)

Primary studies use feature selection or dimensionality reduction techniques to aid the development of effective SCP models. We analyzed these studies to determine the most commonly used methods (Table 4). An analysis of 38 primary studies revealed that 58% of them used either a feature selection or a dimensionality reduction technique. According to Table 4 the most commonly used feature selection technique was Correlation-based Feature Selection (CFS). Apart from the techniques listed in Table 4, other primary studies used several other miscellaneous methods (Best-first search (PS12), Variable Importance (PS15), Rough set analysis (PS21, PS37), Information Gain (PS21, PS37), *t*-test (PS23), Chi-square test (PS37), Genetic Algorithm (PS23, PS37), Metric Violation Score (PS27), Wrapper Method (PS36), Consistency Feature selection (PS37), OneR feature evaluation (PS37)). Apart from feature selection, several studies performed correlation analysis to investigate whether the predictors used are correlated with change-proneness attribute (PS8, PS9, PS10, PS11, PS12, PS13, PS19, PS21, PS22, PS37).

Certain studies in literature reported specific OO metrics as effective predictors of change-prone nature of a class. These metrics were selected after application of feature selection or dimensionality reduction techniques. Since, in RQ1 we reported that the CK metrics suite and the SLOC metric are popular amongst primary studies, we state the studies which report these metrics as effective indicators of change-proneness (Table 5). According to the table, 14 studies reported metrics which characterize size attribute (SLOC and WMC) and the ones which characterize coupling attribute (CBO and RFC) as efficient indicators of change-proneness. Moreover, it may be noted that the inheritance attribute metric, DIT was only selected as an effective metric by three studies and there was no study which selected NOC (another inheritance metric) as an effective predictor of change-proneness.

5.2.2. Dataset characteristics (RQ2.2)

In order to perform empirical validation, primary studies have used a number of datasets. This question explores the characteristics of these datasets which includes their nature (public/private), size, percentage of change and other attributes.

Software datasets used by primary studies can be broadly categorized into public/open-source datasets or private/commercial datasets. We categorized the datasets in SCP studies and found that only 5% of these studies used commercial/private datasets. All other SCP studies used open-source datasets, which are publicly available. This trend was observed as commercial datasets are difficult to obtain and is similar to the one observed by Catal and Diri [10] while

Table 4. Feature selection/dimensionality reduction techniques

Feature selection/dimensionality reduction	Study numbers
Correlation-based Feature Selection (CFS)	PS13, PS18, PS19, PS24, PS25, PS26, PS30, PS34, PS35, PS37, PS38
Univariate Analysis	PS21, PS22, PS24, PS28, PS37
Principal Component Analysis (PCA)	PS12, PS20, PS21, PS37
Gain Ratio	PS21, PS29, PS37
Multivariate Regression with forward and backward selection	PS3, PS13, PS22

Table 5. OO metrics selected for SCP in primary studies

Metric Acronym	OO Attribute	Study Numbers
SLOC	Size	PS4, PS8, PS15, PS18, PS19, PS21, PS24, PS25, PS26, PS28, PS33, PS34, PS37, PS38
WMC	Size	PS8, PS15, PS18, PS20, PS22, PS24, PS25, PS28, PS33, PS34
CBO	Coupling	PS8, PS18, PS20, PS21, PS22, PS24, PS25, PS28, PS33, PS37, PS38
RFC	Coupling	PS8, PS13, PS15, PS19, PS20, PS21, PS28, PS37
LCOM	Cohesion	PS15, PS20, PS21, PS28, PS37
DIT	Inheritance	PS20, PS28, PS37

reviewing defect prediction literature. Therefore, researchers tend to validate their results on datasets that are open-source and easily available in software repositories.

We also investigated the language used to develop the datasets, which are used for empirical validation for SCP. Only four studies (PS1, PS2, PS15, PS18) used datasets developed using the C++ language. It may be noted that all other studies used datasets developed in Java language.

The datasets used in primary studies for SCP are of varying sizes and with different percentage of change-prone classes. For each study, we analyzed the number of datasets used and the minimum and maximum size of datasets in terms of number of data points, i.e. classes (Table 6). We also state the minimum and maximum percentage of change in the datasets used by these studies (Table 6). It was noted that certain datasets were used by more than one primary study. The name of such datasets and the studies which use them are listed in Table A2 (Appendix A).

It is also important to evaluate whether the datasets used for developing models are imbalanced in nature. A dataset is said to be imbalanced if it has a disproportionate number of change-prone and not change-prone classes. We state the number of datasets which were found to be imbalanced for a specific study in Table 6. As it is more important to determine the change-prone classes correctly, one should have sufficient number of change-prone classes in a dataset for effectively training the model. We term a dataset as imbalanced if it has less than 40% of change-prone classes. Studies from which relevant information could not be extracted are not shown in the table. According to the information shown in Table 6, the size of datasets used in primary studies for SCP varies from 18–3,150 data points. Therefore, these studies have analyzed small sized, moderately sized and large-sized datasets for developing SCP models. It may also be noted that the percentage of change found in these datasets varies from 1–97%. However, in a majority of the studies 25–100% of datasets analyzed were imbalanced in nature. Only few studies (PS26, PS30, PS35, PS36) addressed the issue of learning from imbalanced training data in SCP literature. It is mandatory for researchers to take active steps to develop effective prediction models from imbalanced datasets in order to develop reliable and unbiased models. On the other hand, it should be noted that though having an imbalanced dataset is an issue while training the model, it is good

Table 6. Study-wise details of datasets

S.No.	Number of data points		Percentage of change		Number of datasets Imbalanced datasets (%)
	Minimum	Maximum	Minimum	Maximum	
PS1	–	1,211	–	24%	1 (100%)
PS2	–	1,211	–	24%	1 (100%)
PS3	58	169	25%	50%	2 (50%)
PS4	–	58	–	25%	1 (100%)
PS5	18	2,737	–	–	15 (–)
PS6	44	62	–	–	1 (–)
PS7	18	958	–	–	8 (–)
PS8	38	693	–	–	3 (–)
PS9	38	2,845	–	–	102 (–)
PS10	25	165	–	–	10 (–)
PS11	98	788	–	–	2 (–)
PS12	36	170	4%	91%	20 (50%)
PS13	254	657	10%	52%	3 (67%)
PS14	607	2,786	1%	97%	12 (25%)
PS15	108	510	45%	66%	6 (0%)
PS16	–	–	–	–	18 (–)
PS17	36	60	–	–	2 (0%)
PS18	108	510	45%	66%	3 (0%)
PS19	685	756	24%	33%	2 (100%)
PS20	36	170	4%	78%	13 (38%)
PS21	–	–	–	–	1 (–)
PS22	1,507	1,524	7%	16%	5 (100%)
PS23	83	1,943	30%	68%	10 (30%)
PS24	348	434	4%	30%	2 (100%)
PS25	72	374	19%	63%	6 (50%)
PS26	72	350	6%	37%	6 (100%)
PS27	53	3,150	8%	94%	14 (57%)
PS28	608	1,496	9%	46%	5 (80%)
PS29	–	–	19%	35%	20 (100%)
PS30	341	1,505	3%	83%	20 (65%)
PS31	53	3,150	2%	93%	14 (64%)
PS32	86	130	14%	59%	4 (75%)
PS33	210	375	4%	52%	9 (67%)
PS34	78	1,404	29%	88%	10 (40%)
PS35	272	1,705	8%	17%	8 (100%)
PS36	121	2,2,46	22%	37%	33 (100%)
PS37	83	1,943	30%	68%	10 (30%)
PS38	222	1,101	16%	62%	15 (53%)

Note: "–" indicates the corresponding information was not found in the study.

that only few classes are change-prone. Therefore, only these few classes require constant monitoring and most of the maintenance resources can be focused on these classes. In case majority of the classes are change-prone, software practitioners might face a tough time managing constraint resources during maintenance and testing.

5.2.3. Validation methods (RQ2.3)

Studies in literature have used various validation methods for developing SCP models which can be broadly categorized into within-project methods and cross-project methods. Within-project validation models use training and testing data of the same software project. The training data used by the model is obtained from the previous versions of the same project and is validated on the later versions. On the contrary, in cross- project validation, the prediction model is trained using data from one project (say Project A) and is validated on another project (Project B). Cross-project validation is useful in case historical data of the same software project is not available. Figure 2 depicts the most commonly used validation methods in SCP studies. An analysis of the figure reveals that majority of studies developed models with within-project approach. It can be performed using either hold-out validation, K-fold cross validation or Leave-one-out Cross Validation (LOOCV), which are described below:

- LOOCV: For a dataset having N instances, this method requires N iterations. In each iteration, all data points except one are used as training instances. The remaining data point is used for validation. It is ensured that all data points are used at least once for validating the developed model. Only one study (PS17) used LOOCV.
- K-fold Cross Validation: The whole dataset is randomly split into K parts, which are nearly equal in size. Thereafter, K iterations are performed. In each iteration, only one partition is excluded for validation, while all others are used for training the model. As in LOOCV, each partition is used for validation at least once. The most frequently used value for K is 10. Only one primary study each used $K = 20$ (PS22) and $K = 5$ (PS37). It is the most popular method for validating SCP models.
- Hold-out Validation: The available data points are randomly split into testing and training sets using a specific ratio. One of the most common ratio used for partitioning is 75:25. In such a case, 75% of data points are used while training and the remaining 25% of data points are used while validation. However, the method has high variability due to random division of training and test sets. The points which make the training and test sets may affect the performance of the developed model. Only four studies used hold-out validation.

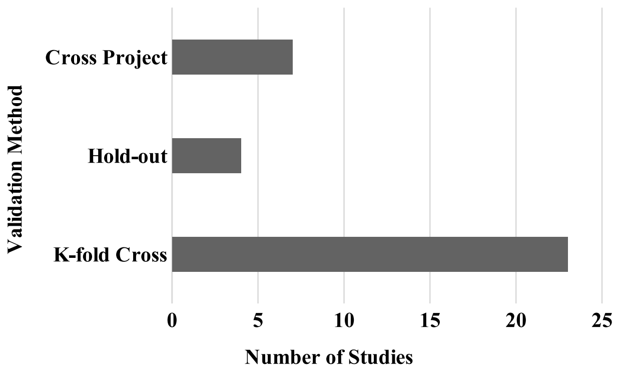


Figure 2. Validation methods in primary studies

Apart from within-project validation, cross-project validation was used by seven SCP studies (PS14, PS18, PS23, PS24, PS30, PS31, PS38). Also, inter-version validation, where different releases of the same dataset are used for training and validation was used by two studies (PS14, PS26). We found that k-fold cross validation is the most popular validation method as it provides the mean results obtained in various partitions, thereby reducing variability. As a result, the data is insensitive to the created partitions as in the case of hold-out validation. PS29 considered the time dimension for validating the developed model. They used a three month sliding window to train and test SCP models as the developers metrics used by the study encapsulate developer dynamics in a given time period.

5.2.4. Performance measures (RQ2.4)

The developed SCP models in primary studies are assessed using various performance measures. This RQ investigates the most commonly used performance measures, depicted in Figure 3. The definitions of these measures are stated as follows:

- Accuracy: It depicts the percentage of correctly predicted classes (change-prone and not change-prone category).
- Recall: It is an estimate of the percentage of correctly predicted change-prone classes amongst the total number of actual change-prone classes. It is also commonly referred to as Sensitivity. A complementary measure of Recall is specificity. Specificity represents the percentage of correctly predicted

not change-prone classes amongst the total number of actual not change-prone classes.

- Precision: It depicts the percentage of correctly predicted change-prone classes amongst the total number of predicted change-prone classes.
- F-measure: It is computed as the harmonic mean of recall and precision.
- Area Under Receiver Operating Characteristic Curve (AUC): It is a plot of recall and specificity. Recall is depicted on the y-axis, while a value of 1 specificity is depicted on the x-axis. The area under the depicted plot gives an estimate of the model's performance.

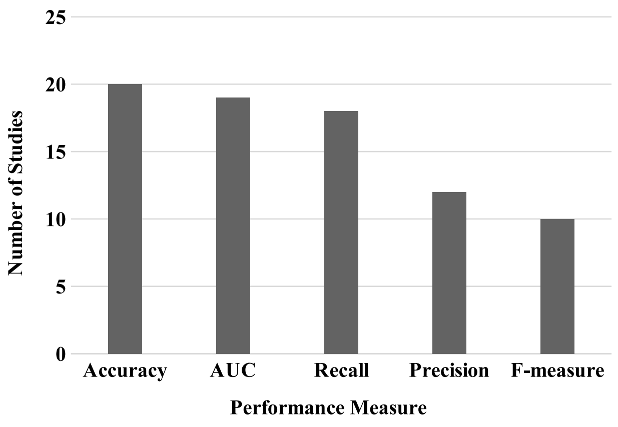


Figure 3. Commonly used performance measures in primary studies

It may be noted that a model which attains higher value for all the discussed performance measures is desirable. According to Figure 3, the most commonly used measure is accuracy. However, in case of imbalanced datasets, accuracy is not an appropriate measure [66–68]. Even if the percentage of correctly predicted change-prone classes are very few, accuracy values can be high as the performance measure is not sensitive to class distributions. On the contrary, the AUC measure is effective as it takes into account both recall and 1 specificity. Researchers should use an appropriate performance measure to yield unbiased results. Selection of an appropriate performance measure is vital to strengthen the conclusion validity of the study. Apart from the measures shown in Figure 3, there were several other performance measures (Type I error, Type II error, Overall misclassification error, False positive ratio, False negative ratio, Specificity, Probability of False Alarm (PF), Goodness of fit, J-index, G-measure, G-mean, Change cost, cost ratio, Balance, Mathews Correlation Coefficient), which were used by only few studies.

5.3. Data analysis algorithms used for SCP (RQ3)

Prediction models require the aid of data analysis algorithms, which can be broadly categorized into statistical or ML. Statistical algorithms include regression techniques such as binary Logistic Regression (LR), polynomial regression or Linear Discriminant Analysis (LDA). ML algorithms include various categories such as Decision Trees (DT), Bayesian algorithms, Artificial Neural Networks (ANN), ensembles, Search-Based Algorithms (SBA), etc. We first investigate the most popular category of algorithms for developing SCP models.

5.3.1. Popular category of data analysis algorithms (RQ3.1)

Certain primary studies used only a specific category of algorithm, i.e. only statistical or only ML, while certain others used more than one category. Figure 4 depicts the number of primary studies using the various categories of algorithms. A new category of algorithms, i.e. ensembles algorithms were used by certain studies (PS17, PS20, PS21, PS23, PS34, PS37, PS38), which were ensemble of several base learning algorithms. For instance, PS17 used an ensemble of Multilayer Perceptron (MLP), Support Vector Machine (SVM), Genetic Programming (GP), Logistic Regression (LR) and k-means techniques which were aggregated using majority voting. According to Figure 4, ML algorithms are the most popular category, followed by the statistical algorithms. The disadvantage of statistical algorithms over ML ones is that the models developed using statistical techniques are not easily interpretable [69]. Another disadvantage of statistical models is that they are highly reliant on data distribution and are based on assumptions which may not be fulfilled by the software product data whose change-proneness is to be predicted [69]. Out of the 38 studies, three studies did not use any specific algorithm but predicted classes using a certain set of equations (PS4), by using a combined rank list (PS8) or by using random effect meta-analysis model (PS9).

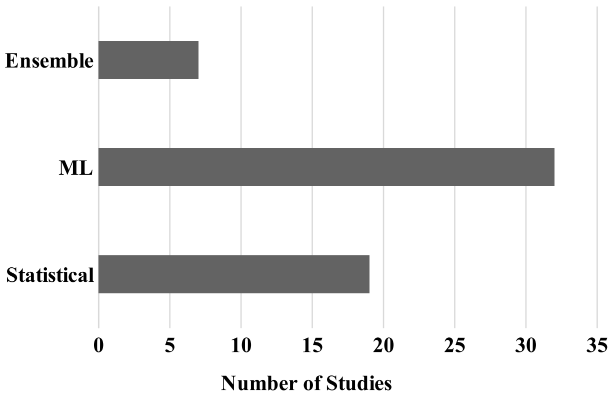


Figure 4. Categories of techniques

5.3.2. ML algorithms used for SCP (RQ3.2)

The ML algorithms can be further divided into several categories in accordance with Malhotra [15]. Table 7 states the various sub-categories of ML techniques which are used by SCP studies. These sub-categories are Decision Trees (DT), Bayesian algorithms, SVM, ML Ensemble, ANN and SBA. Other remaining algorithms were grouped into a miscellaneous category.

We further analyzed the percentage of primary studies which used a specific category of ML algorithms amongst the primary studies which used an ML algorithm for SCP (Figure 5). It was noted that ANN is the most popular category of ML algorithms which are used by 53% of studies. ANN are capable of modeling complex non-linear relationships and are adaptive in nature making them suitable for change prediction tasks. The next popular category of techniques were SBA, used by 41% of studies. It is a subclass of ML algorithms, which have recently gained popularity. SBA are self-optimizing techniques, which are capable of dealing with noisy and imprecise data. ML ensemble algorithms, which form several classification models using variants of training set and use voting scheme to combine these models are also a popular category of techniques used by 41% of studies.

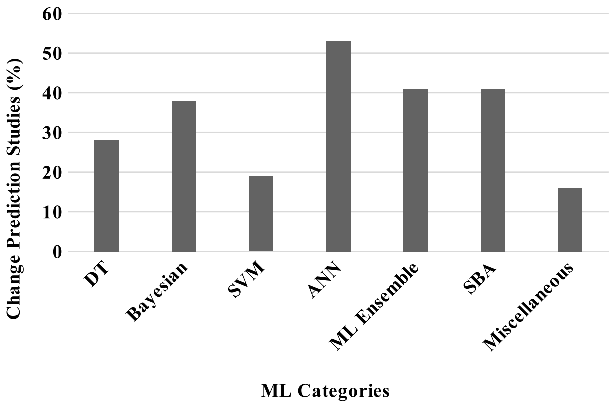


Figure 5. Sub-categories of ML algorithms

5.4. Predictive performance of ML algorithms for SCP (RQ4)

The various ML algorithms investigated in the primary studies for developing SCP models should be assessed so as to ascertain their effectiveness.

5.4.1. Predictive capability of ML algorithms

In order to assess the capability of ML algorithms, we state the values of popular performance measures of the developed SCP models. However, we need to generalize our results and avoid any bias. This was done by reporting the results of models developed by those algorithms which were validated by using at least three different datasets and by at least two of the primary studies. This would forbid an algorithm which exhibits exceptional performance only in a certain study or only by using certain datasets to be declared as a superior one. We analyze the statistics in accordance with the datasets. However, it may be the case that the performance of a technique varies due to its application on a specific dataset. Thus, we remove outlier values in accordance with the investigated datasets. We also report the median values to reduce biased results. The following rules were observed while extracting various statistics [6, 15]. The rules are chosen so that optimum values attained by a technique may be reported. This is

Table 7. Sub-categories of ML algorithms

Sub-category	ML algorithms
Decision Tree (DT)	C4.5, J48, Classification And Regression Tree (CART)
Bayesian	Naive Bayes (NB), Bayesian Network (BN)
SVM	SVM, Linear Kernel SVM, Sigmoid Kernel SVM, Polynomial Kernel SVM, Least-Square SVM (Linear, Polynomial and RBF kernels)
Artificial Neural Networks (ANN)	MLP, MLP with Conjugate Learning (MLP-CG), Radial Basis Function (RBF), Group Method of Data Handling (GMDH), Extreme ML (Linear, Polynomial and RBF kernels)
ML Ensemble	Random Forests (RF), Bagging (BG), Adaptive Boosting (AB), LogitBoost (LB)
Search Based Algorithms (SBA)	Ant Colony Optimization (ACO), Constricted Particle Swarm Optimization (CPSO), Decision Tree-GP, Decision Tree- GA (DT-GA), GP, Genetic Expression Programming (GEP), Hierarchical Decision Rules (HIDER), Memetic Pittsburgh Learning Classifier System (MPLCS), Supervised Classifier System (SUCS), X Classifier System (XCS), Genetic Algorithm with Adaptive Discretization Intervals (GA-ADI), Fuzzy Learning based on Genetic Programming Grammar Operators and Simulated Annealing (GFS-SP), Fuzzy Learning based on Genetic Programming (GFS-GP), Genetic Fuzzy System AdaBoost (GFS-AB), Genetic Fuzzy System- LogitBoost (GFS-LB), Genetic Fuzzy System MaxLogitBoost (GFS-MLB), Genetic Algorithm with Neural Networks (GANN), Neural Net Evolutionary Programming (NNEP), Particle Swarm Optimization- Linear Discriminant Analysis (PSO-LDA); Structural Learning Algorithm in a Vague Environment with Feature Selection (SLAVE)
Miscellaneous	K-Nearest Neighbor (K-NN), k-means, KStar, Non-Nested Generalized Exemplars (NNGE), PART, Decision Table

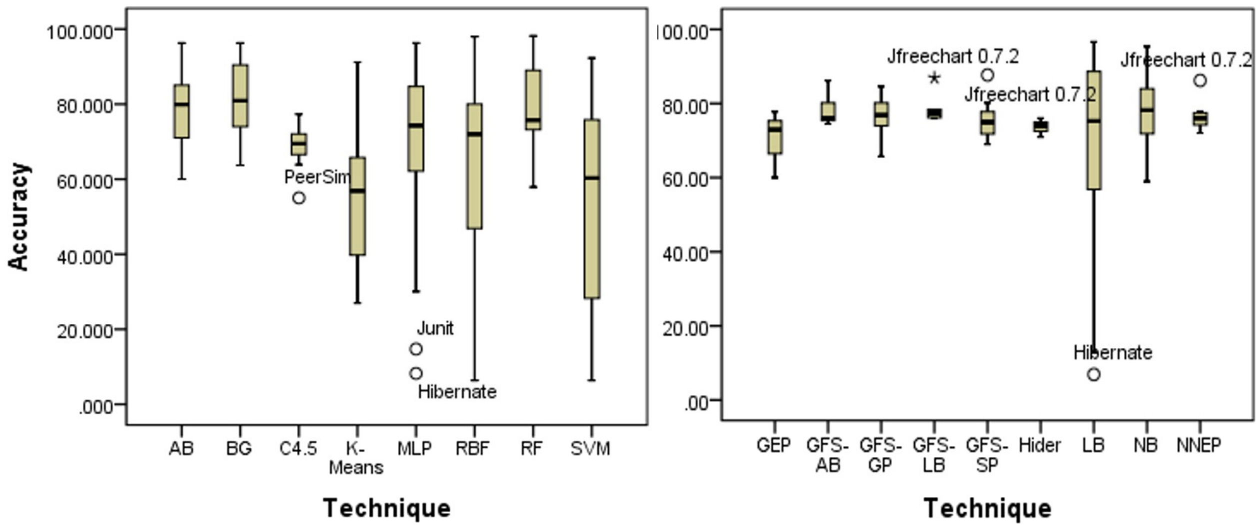


Figure 6. Dataset-wise accuracy outliers of ML algorithms

important as the performance of an ML algorithm is dependent on its internal parameter settings.

– If a specific study develops models on the same dataset more than once with different experimental settings, we choose the best performance measure values obtained by the technique.
– In case there is more than one study which develops models using the same dataset and the same technique, we use the best of per-

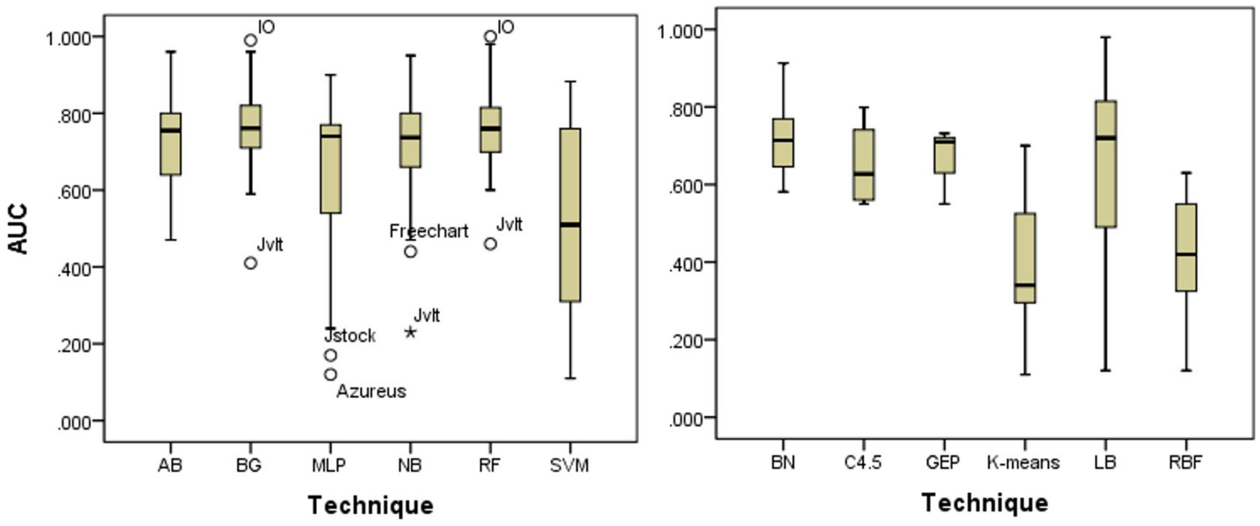


Figure 7. Dataset-wise AUC outliers of ML algorithms

formance measure value reported in all the studies.

According to Section 5.2.4, the most commonly used performance measures by SCP studies are accuracy and AUC. Figure 6 depicts the dataset-wise outliers of different ML algorithms with respect to accuracy measure. According to the figure, the Hibernate dataset was an outlier for both MLP and LB algorithms, showing lower accuracy values than all other investigated datasets. JFreechart 0.7.2 is exhibited as an outlier for GFS-LB, GFS-SP and NNEP algorithms. Figure 7 depicts the dataset-wise outliers of different ML algorithms with respect to AUC measure. According to the figure, NB, BG, RF and MLP algorithms were found to have outliers with lower AUC values except the IO dataset which had higher AUC values for BG and RF.

As discussed before, a good change prediction model exhibits higher values of accuracy and AUC measures. Table 8, 9 presents the comparative results of the change prediction models developed using ML algorithms for the accuracy and AUC measure respectively. The tables report the statistics values along with the count of datasets from which the statistics were extracted, after removing the outliers.

As depicted in Table 8, the majority of ML techniques (except k-means and SVM) depicted mean accuracy values in the range 60–80%. The BG technique depicted the best mean accuracy value of 81.72%. With respect to median accuracy values, the best median value was depicted by the BG technique. As depicted in Table 9, with respect to AUC, the majority of ML techniques (except k-means, RBF and SVM) depicted a mean AUC value in the range 0.65–0.78. Both the BG and RF techniques depicted the highest mean AUC value of 0.77. The best median AUC values were depicted by AB, RF and BG techniques of 0.76 each. These results indicate effectiveness of ML techniques in determining change-prone nature of classes/modules.

It may be noted that the BG, RF and AB techniques belong to the ensemble category of ML algorithms. Therefore, their effective predictive capability is a result of aggregation of results of several base models. This leads to stable and robust models. It may also be noted that the SBA (GFS-AB, GFS-GP, GFS-LB, GFS-SP, NNEP, HIDER) also exhibit good accuracy results. SBA are effective in optimizing the accuracy of the developed SCP models. This category of ML algorithms needs to be further explored as their results are promising. The statistics reported in Tables 8 and 9 reveal that the use of ML algorithms for change-proneness prediction tasks should be encouraged as they yield effective results.

Furthermore, we also conducted a meta-analysis of the review studies, which reported the AUC

Table 8. Accuracy results of ML algorithms for SCP models

ML Algorithm	Count	Minimum	Maximum	Mean	Median	SD
AB	17	60.00	96.30	78.92	79.90	10.78
BG	16	63.71	96.30	81.72	80.95	9.25
C4.5	11	63.86	77.33	69.95	69.99	3.55
GEP	3	60.00	77.78	70.24	72.94	7.50
GFS-AB	6	74.50	86.20	78.05	76.00	4.06
GFS-GP	6	65.70	84.60	76.38	76.90	5.91
GFS-LB	5	76.00	78.40	77.10	77.00	0.93
GFS-SP	6	69.00	80.20	73.87	73.50	3.56
HIDER	3	71.00	76.00	73.67	74.00	2.05
k-means	16	26.99	91.17	54.51	56.91	18.73
LB	28	12.88	96.61	71.13	78.10	22.75
MLP	38	30.10	96.30	73.00	75.72	16.56
NB	15	59.00	95.38	77.98	78.19	10.22
NNEP	6	72.00	77.90	75.25	75.25	1.98
RF	25	57.91	98.18	79.32	75.70	10.58
RBF	27	6.38	98.00	63.32	72.00	24.69
SVM	17	6.38	92.29	53.15	60.33	28.20

SD indicates Standard Deviation.

Table 9. AUC results of ML algorithms for SCP models

ML Algorithm	Count	Minimum	Maximum	Mean	Median	SD
AB	25	0.47	0.96	0.74	0.76	0.18
BG	32	0.59	0.96	0.77	0.76	0.11
BN	21	0.58	0.91	0.72	0.71	0.09
C4.5	4	0.55	0.80	0.65	0.63	0.09
GEP	3	0.55	0.73	0.66	0.71	0.08
k-means	16	0.11	0.70	0.40	0.34	0.17
LB	31	0.12	0.98	0.65	0.72	0.23
MLP	47	0.24	0.90	0.69	0.74	0.16
NB	32	0.47	0.95	0.74	0.75	0.09
RBF	16	0.12	0.63	0.41	0.42	0.16
RF	31	0.60	0.98	0.77	0.76	0.10
SVM	27	0.11	0.88	0.53	0.51	0.24

SD indicates Standard Deviation.

performance measure. This was done in order to evaluate the performance of ML methods in the domain of SCP. Figure 8 reports the forest plot of primary studies with the summary performance measure statistic per study. The weights to the primary study were allocated on the basis of standard error, i.e. higher standard error indicated lower study weight [70]. The confidence interval is computed at 95%. We assessed the random effects model as the studies were heterogeneous in terms of datasets, ML methods and their performance. The overall effect in the figure indicates that ML methods are effective for SCP.

5.4.2. Comparative performance of ML algorithms

We investigate the comparative performance of various ML algorithms with each other and with traditional statistical algorithms used for developing SCP models. The explored hypothesis is stated as follows:

Null Hypothesis (H0): There is no statistical difference amongst the performance of different ML algorithms when compared with each other and with the statistical technique (LR), while developing SCP models.

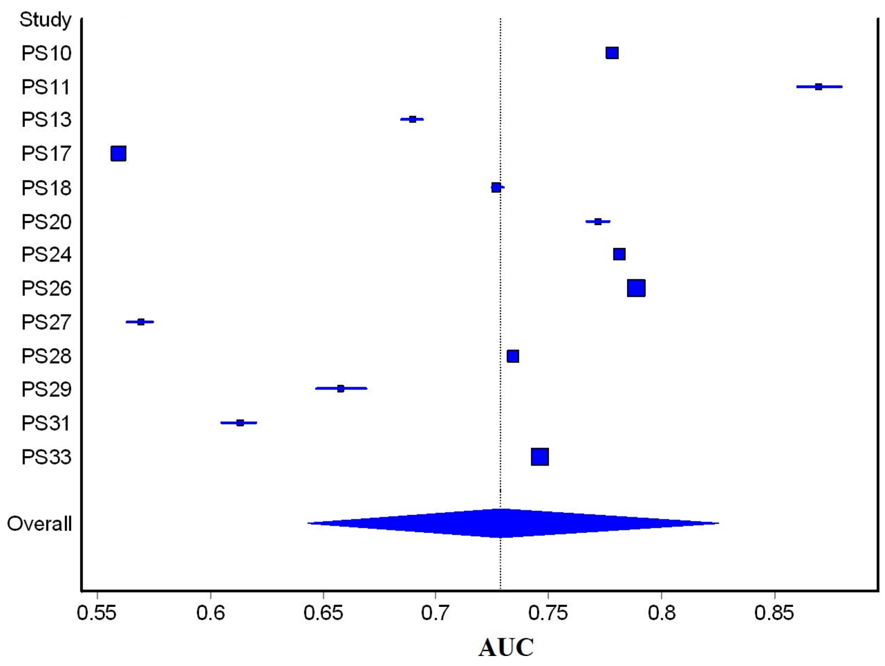


Figure 8. Forest plot

Alternate Hypothesis (H1): There is significant difference amongst the performance of different ML algorithms when compared with each other and with the statistical technique (LR), while developing SCP models.

The comparative performance was evaluated dataset-wise and the rules were similar to the ones followed in RQ4. Furthermore, Wilcoxon signed rank test was performed at a significance level of 0.05 for statistical evaluation of the comparative results. We compared the performance of 17 ML algorithms namely, MLP, BG, AB, RF, RBF, SVM, C4.5, *k*-means, LB, HIDER, GFS-AB, GFS-GP, GFS-LB, GFS-SP, NNEP, NB and BN amongst each other and with LR. LR is chosen as it is the most common statistical algorithm used in SCP literature. The other ML algorithms were chosen as we could extract sufficient data from primary studies for their comparison.

Tables 10 and 11 report the results of the Wilcoxon signed rank test when different algorithms are compared with one another and with the LR algorithm according to accuracy and AUC performance measures respectively. The symbols used in the table represent whether the performance of the technique stated in the row is significantly superior (BT*), significantly inferior (WR*), superior but not significantly (BT), inferior but not significantly (WR) or equivalent (=), when compared with the technique stated in the column. We consider the two compared techniques as equivalent when the pairwise comparison amongst the techniques yield equal number of negative and positive ranks in Wilcoxon signed rank test. According to Table 10, the MLP technique shows significantly better performance than LR, C4.5 and NB techniques in terms of accuracy measure. The performance of MLP technique is worse but not significantly, when compared with the AB technique. MLP's accuracy performance is better than RF, RBF, SVM, *k*-means and LB techniques but not significantly. The Wilcoxon test results according to AUC measure depicted in Table 11 show that the RF, LB, BN and NB algorithms showed significantly better AUC performance than various other algorithms. The MLP algorithm also depicts better AUC values than five other compared algorithms, but not significantly.

It may be noted from the results of Tables 10 and 11 that three ML algorithms depicted better accuracy results than the statistical algorithm, LR. However, four algorithms (SVM, GFS-SP, NNEP and AB) showed worse accuracy results than LR. With respect to AUC, five ML

Table 10. Comparison of ML algorithms based on accuracy measure (Wilcoxon test results)

Algo.	MLP	BG	AB	RF	LR	RBF	SVM	C4.5	KM	LB	HIDER	GAB	GGP	GLB	GSP	NNEP	NB
MLP	–	WR*	WR	BT	BT*	BT	BT	BT*	BT	BT	ND	ND	ND	ND	ND	ND	BT*
BG	BT*	–	BT	WR	BT	ND	ND	ND	ND	WR	ND	ND	ND	ND	ND	ND	BT
AB	BT	WR	–	WR	WR	ND	ND	ND	ND	WR	WR	ND	ND	ND	WR	WR	BT*
RF	WR	BT	BT	–	BT*	WR	ND	BT*	ND	BT	ND	ND	ND	ND	ND	ND	BT*
RBF	WR	ND	ND	BT	ND	–	BT	BT*	WR	WR	ND	ND	ND	ND	ND	ND	ND
SVM	WR	ND	ND	ND	WR	ND	–	ND	WR	WR*	ND	ND	ND	ND	ND	ND	ND
C4.5	WR*	ND	ND	WR*	ND	WR*	ND	–	ND	ND	ND	ND	ND	ND	ND	ND	ND
k-means	WR	ND	ND	ND	ND	BT	BT	ND	–	WR	ND	ND	ND	ND	ND	ND	ND
LB	WR	BT	BT	WR	ND	BT	BT*	ND	BT	–	ND	ND	ND	ND	ND	ND	BT*
HIDER	ND	ND	BT	ND	ND	ND	ND	ND	ND	ND	–	ND	ND	ND	BT	WR	ND
GFS-AB	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	–	BT	WR	BT	BT	ND
GFS-GP	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	WR	–	WR	WR	WR	ND
GFS-LB	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	BT	BT	–	BT	BT	ND
GFS-SP	ND	ND	BT	ND	WR	ND	ND	ND	ND	ND	WR	WR	BT	WR	–	WR	ND
NNEP	ND	ND	BT	ND	WR	ND	ND	ND	ND	ND	BT	WR	BT	WR	BT	–	ND
NB	WR*	WR	WR*	WR*	ND	ND	ND	ND	ND	WR*	ND	ND	ND	ND	ND	ND	–

GAB: GFS-AB; GSP: GFS-SP; GLB:GFS-LB; GGP:GFS-GP; KM: *k*-means; "BT*": Significantly better results; "BT": Better but insignificant; "WR*" indicates significantly worse results and "WR" means worse but not significant results;
"ND" indicates requisite comparison data could not be extracted; "–" indicates the techniques cannot be compared with itself.

Table 11. Comparison of ML algorithms based on AUC measure (Wilcoxon test results)

Technique	MLP	NB	SVM	BG	AB	RF	LR	LB	RBF	C4.5	*k*-means	BN
MLP	–	BT	BT	WR*	WR	WR*	WR*	WR*	BT	BT	BT	WR*
NB	WR	–	BT	BT*	BT*	WR*	BT*	BT*	ND	ND	ND	BT
SVM	WR	WR	–	WR	WR	WR	WR*	WR*	WR	ND	WR	WR*
BG	BT*	WR*	BT	–	BT*	WR	BT	BT	BT	ND	ND	BT
AB	BT	WR*	BT	WR*	–	WR	BT	WR	BT	ND	ND	WR
RF	BT*	BT*	BT	BT	BT	–	BT	BT	BT	ND	ND	WR
LB	BT*	WR*	BT*	WR	BT	WR	WR	–	BT	ND	BT	WR*
RBF	WR	ND	BT	WR	WR	WR	ND	WR	–	ND	WR	WR*
C4.5	WR	ND	ND	ND	ND	ND	ND	ND	ND	–	ND	BT
k-means	WR	ND	BT	ND	ND	ND	ND	WR	BT	ND	–	WR*
BN	BT*	WR	BT*	WR	BT	BT	BT	BT*	BT*	ND	BT*	–

Symbols same as Table 10.

techniques were found better than LR and three were found worse than LR. This indicates effective performance of ML algorithms when compared to that of the LR algorithm. However, more studies need to be conducted for an extensive comparison of various ML algorithms with that of LR. Also, as a number of columns in Tables 10 and 11 have the value "ND", where sufficient data was not found to compare the predictive ability of ML algorithms. Therefore, more studies are needed which perform extensive comparison of different ML algorithms with each other based on different performance measures. However, on the basis of current analysis we reject the null hypothesis H0.

5.5. Statistical tests used by SCP studies (RQ5)

Statistical verification of a study's results is important in order to yield reliable conclusions. 66% of primary studies used statistical tests for validating their results. These tests can be broadly categorized as parametric tests or non-parametric tests.

Twenty-five primary studies which predicted change-prone nature of a class/module statistically validated their results. Out of these 25 studies, 88% of studies used non-parametric tests, while the others used parametric tests. This trend was observed as parametric tests require stringent assumptions which should be fulfilled before their application. In order to verify the assumptions of normal tests, we require complete information about the population distribution. Though these characteristics make normal tests powerful but they are harder to apply when compared with non-parametric tests. Non-parametric tests are easy to understand and use. Thus, they are favored by the research community. Figure 9 states the number of studies using the most commonly used statistical tests. These tests were the Wilcoxon signed rank test, Friedman test, *t*-test, Scott–Knott test and Cliff's test used by 15, 9, 2, 2 and 2 studies, respectively. The most popular test was Wilcoxon signed rank test. The popularity of Wilcoxon test is due to its non-parametric nature. Moreover, the test can be used individually for pairwise comparisons or as a post-hoc test after the application of Friedman test [6]. Certain other tests (ANOVA, Mann–Whitney, Nemenyi, Proportion) were used by one study each.

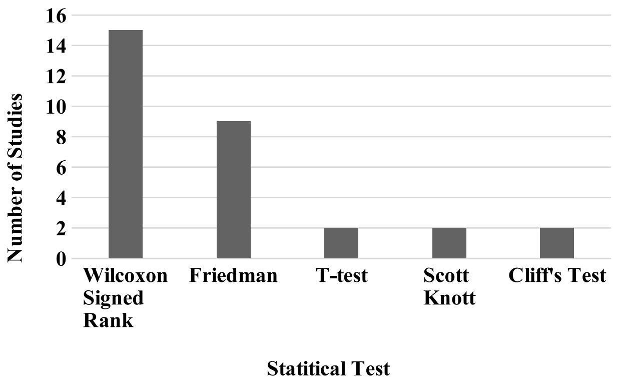


Figure 9. Statistical tests used in primary studies

5.6. Threats to validity in SCP studies (RQ6)

This RQ extracts and analyzes the threats to empirical studies which develop SCP models. It is essential for a researcher to scrutinize all probable threats in the early phases of an experiment so that the obtained results are valid and reliable. This would ensure proper experimental design so that majority of identified threats can be mitigated. Furthermore, one should mention the probable threats so that the readers are aware of the limitations. We extracted the threats from the primary studies of the review, which have a separate section for "Threats to validity" or "Limitations".

5.6.1. Categories of threats (RQ6.1)

The probable threats to SCP studies are categorized into conclusion, internal, construct and external threats.

Table 12 states the various threats corresponding to each category along with the studies which state them. It may be noted that we state only those threats which are mentioned in at least two or more primary studies. Threats specific to a study's experimental design are omitted to

Table 12. Threats to validity in SCP studies

Threat No.	Category	Threat Description (Study Numbers)
T1	Conclusion	Absence of appropriate statistical tests for validating study's results (PS10, PS16, PS25, PS26, PS29, PS33, PS34, PS36, PS38).
T2	Conclusion	Absence of multiple and stable performance measures (PS25, PS26, PS29, PS33, PS36, PS38).
T3	Conclusion	Not accounting for validation bias by using inappropriate validation method (PS25, PS26, PS29, PS33, PS34, PS38).
T4	Internal	Omittance of significant variables that act as predictors or may affect the predictors (PS3, PS11, PS31).
T5	Internal	Inability to address the confounding effect of other variables such as class size or other factors (such as developer experience, application domain, etc.) on the relationship between dependent and independent variables (PS9, PS12, PS27, PS33).
T6	Internal	Does not account for the "causal effect" of the predictors on the target variable (PS11, PS12, PS13, PS19, PS24, PS25, PS26, PS32).
T7	Internal	Does not account for different rules or thresholds for computing the dependent and the independent variables (PS9, PS10, PS24, PS27, PS29).
T8	Construct	The type of change, i.e. whether it is corrective, adaptive, perfective or preventive is not taken into account (PS9, PS12, PS13, PS19, PS24, PS29, PS36, PS38).
T9	Construct	OO metrics may not be accurate representatives of the OO concepts they propose to measure (PS9, PS12, PS19, PS24, PS25, PS26, PS33, PS34, PS38).
T10	Construct	Independent variables (OO metrics) and dependent variable may not be correctly collected (PS9, PS11, PS12, PS13, PS16, PS19, PS21, PS24, PS25, PS26, PS32, PS34).
T11	Construct	There may be possible imprecisions in computation of change-proneness attribute (PS29, PS32, PS36, PS38).
T12	Construct	The severity of change and the effort spent by software practitioners in changing code fragment is not taken into account while computing change-proneness (PS29, PS21, PS36).
T13	Construct	Absence of data pre-processing to eliminate noisy data or feature selection for choosing effective feature sets (PS11, PS26, PS29, PS33, PS36).
T14	External	Obtained results may be specific to a certain domain, i.e. all validated datasets belonging to the same domain (PS3, PS10, PS11, PS12, PS19, PS21, PS25, PS26, PS29, PS34, PS35, PS36, PS38).
T15	External	Obtained results may not be validated on datasets of appropriate size or appropriate number of datasets (PS10, PS12 PS13, PS16, PS19, PS22, PS24, PS26, PS27, PS29, PS31 PS32, PS34, PS35, PS36, PS38).
T16	External	Obtained results may not be easily replicated (PS10, PS16, PS25, PS26, PS33, PS34, PS38).
T17	External	Obtained results may not be validated on industrial datasets (PS16, PS22, PS31).
T18	External	Obtained results may not be validated on datasets developed using different programming languages or programming paradigms (PS13, PS16, PS21, PS24, PS25, PS26, PS27, PS29, PS31, PS32, PS33, PS34, PS36, PS38).

yield unbiased results. As stated in Table 12, we found 3 conclusion validity threats, 4 threats to internal validity, 6 construct validity threats and 5 external validity threats. It may be noted that T16 was also referred to as "Reliability threat" in two studies.

5.6.2. Mitigation of threats (RQ6.2)

This RQ explores how the various threats identified in RQ7.1 are addressed by the primary studies. We state the steps suggested by primary studies to mitigate the corresponding threats

Table 13. Mitigation of threats to validity in SCP studies

Threat No.	Threat Mitigation
T1	The results of a study should be validated using proper statistical tests. In case the underlying data does not fulfill the assumptions of a parametric statistical test, non-parametric statistical tests may be used.
T2	Multiple and stable performance measures should be used which give a realistic estimate of the model's performance.
T3	One should use an appropriate validation method so that the results are not biased due to selection of training and testing datasets.
T5	The confounding effect of variables may be evaluated by first building a univariate regression model of the confounding variable C on each predictor P. Thereafter, find the difference between predicted values by the regression model from P to obtain a new variable P'. The obtained P' is free from confounding effect.
T6	Controlled experiments should be carried out where only one specific predictor variable should be varied while keeping all other variables constant to determine the "causal" effect of predictor variables.
T7	Additional thresholds or rules may be used to determine the impact of these on dependent and the independent variables.
T9	OO metrics which are commonly used in literature and have been validated by previous studies may be used.
T10	The tools used for collecting independent and dependent variables should be manually verified to ascertain their correctness. The use of public datasets, which have been verified by previous studies also mitigate the threat.
T11	Strategies which have been well recognized in previous literature studies for computation of change-proneness attribute should be adopted, i.e. designation of a class as change-prone or not change-prone should be done in accordance with the definitions that have been well established in the past such as those followed by [9, 12].
T13	Effective data pre-processing strategies should be adopted for eliminating noisy data. Moreover, selection or extraction methods should be used for selecting relevant features.
T14	The results should be validated on datasets belonging to different domains.
T15	The results should be validated on datasets of appropriate size and on an appropriate number of datasets.
T16	The use of open-source datasets enhances the replicability of the study. Furthermore, the tools used to implement the approach should be available. The steps conducted in the experiment should be clearly presented to ease replicated experiments.
T17	The results should be validated on industrial datasets or datasets whose characteristics are similar to industrial datasets.
T18	The results should be validated on datasets developed using different programming languages.

in Table 13. The table states the mitigation of only those threats, whose mitigation could be extracted from primary studies.

The threats which were only mentioned in the "Threats to Validity" section of primary studies (T4, T8, T12), but could not be mitigated by the study or whose mitigation was not suggested are not stated in the table. Researchers should incorporate these steps (Table 13), while designing the experimental set-ups of their study in order to ensure reliable results. Also, several studies should be performed with different size, category, domain and other dataset characteristics to obtain generalized results in the domain of SCP.

6. Threats to validity

While searching for relevant candidate studies, we applied the search string to only the titles of the studies. Thus, we may fail to include studies which do not use the key terms in their titles. However, as we have extensively searched for candidate studies in the mentioned search databases, have included journal as well as conference studies, have searched for key authors and have also searched the reference lists of the included papers, we are positive that we have not missed a relevant study. It may be noted that the review is based on the presumption that all the primary studies are unprejudiced. In case, this is not true, there is a possible threat to the review results [6, 71] The review also rules out all unpublished results [6].

In order to extract primary studies from candidate studies, both the authors independently applied the quality assessment criteria on each study. This practice ensures conclusion validity of the obtained results. Publication bias is a possible threat to the results of this review. In lieu of publication bias, it is highly likely that a primary study would publish positive results on application of a ML technique for developing SCP model as compared to negative results [72]. It could also be a scenario where researchers might claim that their proposed technique outperforms other established techniques in literature. This could lead to an exaggeration of the capability of ML techniques for developing SCP models. This threat was addressed in two ways. Firstly, we included primary studies which "reported and compared various data analysis algorithms amongst themselves for developing SCP models". These studies are unlikely to be biased towards specific ML techniques as they do not propose a data analysis algorithm of their own. Secondly, while comparing the predictive capability of ML techniques, we compared only those techniques which were "validated on at least three different datasets and were used by at least two primary studies" to avoid bias. Also, the statistics reported in the review were dataset wise after removal of outliers. Furthermore, we state the median values to get a realistic estimate of the capability of ML techniques for developing SCP models.

While evaluating the predictive capability of ML techniques, we also state the AUC results apart from accuracy results. Thus, we have accounted for possible bias which could occur by using imbalanced data as AUC is a stable performance measure.

In order to statistically compare the performance of ML techniques for developing SCP models (Tables 9 and 10), we have conducted several tests. However, certain erroneous inferences may occur due to conduct of several tests on the same data. This threat exists in the study.

7. Conclusions and future guidelines

An extensive systematic review was performed to analyze the current state of existing literature in the domain of SCP and to further identify research gaps in this domain. 38 primary studies were chosen to answer the various RQs. In lieu of the result discussions with respect to the explored RQs, we suggest certain guidelines to researchers in the SCP domain which are mentioned below.

- The product metrics especially the CK metrics suite have been widely used in primary studies for developing SCP models. However, the validation of process metrics and their combination with product metrics is limited in this domain. Researchers should conduct studies to assess the capability of only process metrics as well as a combination of both process and product metrics as predictors of software change.
- Feature selection/dimensionality reduction techniques have been used by a majority of studies. However, more studies should examine effective predictors using feature selection techniques in order to develop efficient SCP models.
- Most of the datasets used by the primary studies were open-source in nature. However, more studies should be conducted to validate commercial datasets to yield practical and generalized results. Also, datasets developed using other languages such as C#, Python, etc. needs to be evaluated by literature studies.
- It was observed that 25–100% of datasets in a majority of the SCP studies were imbalanced in nature (had less than 40% of changed classes). Researchers in future should evaluate methods to develop effective models from imbalanced datasets as correct identification of change-prone classes is crucial. This would aid developers in prioritizing their resources effectively during maintenance and testing phases of a software development lifecycle.

– Within project validation has become a common standard while validating SCP models. Though, cross-project validation has also been investigated, however, studies in the future should explore cross-organization and cross-company validation. Effective transfer learning in cross-organization and cross-company scenario is the need of the hour, which should be actively investigated by researchers in future studies. Furthermore, temporal validation, which takes into account the time dimension should also be explored in the SCP domain.
– Apart from accuracy, the use of AUC measure is prominent in literature for evaluating SCP models. Stable performance measures such as AUC should be used by researchers in future as they give a realistic estimate of the performance of models which are developed from imbalanced datasets.
– It was observed that a majority of studies used ML algorithms and these algorithms are effective in the domain of SCP. However, more studies should be conducted which assess and compare the effectiveness of statistical and ML algorithms for SCP as we could find limited data in literature which compares the performance of different algorithms for developing effective SCP models. Also, more researchers should explore the use of ensemble of algorithms as an alternative to other data analysis algorithms for developing SCP models.
– It was found that SBA (HIDER, GFS-AB, GFS-GP, GFS-LB, GFS-SP and NNEP) exhibited effective accuracy results. However, data to assess and compare the ability of SBA's was limited. More studies which investigate the effectiveness of SBA in the domain of SCP are required to yield conclusive results about their capability. Studies should be conducted to evaluate the effectiveness of SBA and compare their performance with other established ML and statistical techniques.
– The results indicate that a majority (66%) of primary studies use statistical tests for verifying the obtained results. This is a good practice which should be continued in future studies.
– It is mandatory to account for possible "Threats to Validity", while designing experiments to yield effective and reliable results.

Though there are a number of research papers that illustrate quantitative results from SCP in lab environments, there is a need for longitudinal studies with developers in industry that focuses on qualitative research so that the effectiveness of SCP models in industry may be understood in depth.

References

[1] A.G. Koru and H. Liu, "Identifying and characterizing change-prone classes in two large-scale open-source products," *Journal of Systems and Software*, Vol. 80, No. 1, 2007, pp. 63–73.

[2] Y. Zhou, H. Leung, and B. Xu, "Examining the potentially confounding effect of class size on the associations between object-oriented metrics and change-proneness," *IEEE Transactions on Software Engineering*, Vol. 35, No. 5, 2009, pp. 607–623.

[3] A.G. Koru and J. Tian, "Comparing high-change modules and modules with the highest measurement values in two large-scale open-source products," *IEEE Transactions on Software Engineering*, Vol. 31, No. 8, 2005, pp. 625–642.

[4] E. Arisholm, L.C. Briand, and A. Foyen, "Dynamic coupling measurement for object-oriented software," *IEEE Transactions on software engineering*, Vol. 30, No. 8, 2004, pp. 491–506.

[5] B.A. Kitchenham, D. Budgen, and P. Brereton, *Evidence-based software engineering and systematic reviews*. CRC Press, 2015, Vol. 4.

[6] R. Malhotra, M. Khanna, and R.R. Raje, "On the application of search-based techniques for software engineering predictive modeling: A systematic review and future directions," *Swarm and Evolutionary Computation*, Vol. 32, 2017, pp. 85–109.

[7] R. Malhotra and M. Khanna, "Threats to validity in search-based predictive modelling for software engineering," *IET Software*, Vol. 12, No. 4, 2018, pp. 293–305.

[8] D. Godara and R. Singh, "A review of studies on change proneness prediction in object oriented software," *International Journal of Computer Applications*, Vol. 105, No. 3, 2014, pp. 35–41.

[9] R. Malhotra and A.J. Bansal, "Software change prediction: A literature review," *International Journal of Computer Applications in Technology*, Vol. 54, No. 4, 2016, pp. 240–256.

[10] C. Catal and B. Diri, "A systematic review of software fault prediction studies," *Expert systems with applications*, Vol. 36, No. 4, 2009, pp. 7346–7354.

[11] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *IEEE Transactions on Software Engineering*, Vol. 38, No. 6, 2011, pp. 1276–1304.

[12] D. Radjenović, M. Heričko, R. Torkar, and A. Živkovič, "Software fault prediction metrics: A systematic literature review," *Information and Software Technology*, Vol. 55, No. 8, 2013, pp. 1397–1418.

[13] R.S. Wahono, "A systematic literature review of software defect prediction: research trends, datasets, methods and frameworks," *Journal of Software Engineering*, Vol. 1, No. 1, 2015, pp. 1–16.

[14] S. Hosseini, B. Turhan, and D. Gunarathna, "A systematic literature review and meta-analysis on cross project defect prediction," *IEEE Transactions on Software Engineering*, Vol. 45, No. 2, 2017, pp. 111–147.

[15] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," *Applied Soft Computing*, Vol. 27, 2015, pp. 504–518.

[16] P.K. Singh, D. Agarwal, and A. Gupta, "A systematic review on software defect prediction," in *2nd International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, 2015, pp. 1793–1797.

[17] C. Catal, "Software fault prediction: A literature review and current trends," *Expert systems with applications*, Vol. 38, No. 4, 2011, pp. 4626–4636.

[18] X. Zhu, Y. He, L. Cheng, X. Jia, and L. Zhu, "Software change-proneness prediction through combination of bagging and resampling methods," *Journal of Software: Evolution and Process*, Vol. 30, No. 12, 2018, p. e2111.

[19] G. Catolino and F. Ferrucci, "An extensive evaluation of ensemble techniques for software change prediction," *Journal of Software: Evolution and Process*, 2019, p. e2156.

[20] G. Catolino, F. Palomba, A. De Lucia, F. Ferrucci, and A. Zaidman, "Enhancing change prediction models using developer-related factors," *Journal of Systems and Software*, Vol. 143, 2018, pp. 14–28.

[21] R. Malhotra and M. Khanna, "Dynamic selection of fitness function for software change prediction using particle swarm optimization," *Information and Software Technology*, Vol. 112, 2019, pp. 51–67.

[22] L. Kumar, S. Lal, A. Goyal, and N. Murthy, "Change-proneness of object-oriented software using combination of feature selection techniques and ensemble learning techniques," in *Proceedings of the 12th Innovations on Software Engineering Conference*. ACM, 2019, p. 8.

[23] Y. Ge, M. Chen, C. Liu, F. Chen, S. Huang, and H. Wang, "Deep metric learning for software change-proneness prediction," in *International Conference on Intelligent Science and Big Data Engineering*. Springer, 2018, pp. 287–300.

[24] H. Lu, Y. Zhou, B. Xu, H. Leung, and L. Chen, "The ability of object-oriented metrics to predict change-proneness: A meta-analysis," *Empirical software engineering*, Vol. 17, No. 3, 2012, pp. 200–242.

[25] M.O. Elish and M. Al-Rahman Al-Khiaty, "A suite of metrics for quantifying historical changes to predict future change-prone classes in object-oriented software," *Journal of Software: Evolution and Process*, Vol. 25, No. 5, 2013, pp. 407–437.

[26] R. Malhotra and M. Khanna, "An exploratory study for software change prediction in object-oriented systems using hybridized techniques," *Automated Software Engineering*, Vol. 24, No. 3, 2017, pp. 673–717.

[27] D. Romano and M. Pinzger, "Using source code metrics to predict change-prone java interfaces," in *27th International Conference on Software Maintenance (ICSM)*. IEEE, 2011, pp. 303–312.

[28] E. Giger, M. Pinzger, and H.C. Gall, "Can we predict types of code changes? An empirical analysis," in *9th Working Conference on Mining Software Repositories (MSR)*. IEEE, 2012, pp. 217–226.

[29] D. Azar and J. Vybihal, "An ant colony optimization algorithm to improve software quality prediction models: Case of class stability," *Information and Software Technology*, Vol. 53, No. 4, 2011, pp. 388–393.

[30] S. Karus and M. Dumas, "Code churn estimation using organisational and code metrics: An experimental comparison," *Information and Software Technology*, Vol. 54, No. 2, 2012, pp. 203–211.

[31] J.M. Bieman, G. Straw, H. Wang, P.W. Munger, and R.T. Alexander, "Design patterns and change proneness: An examination of five evolving systems," in *Proceedings. 5th International*

Workshop on Enterprise Networking and Computing in Healthcare Industry (IEEE Cat. No. 03EX717). IEEE, 2004, pp. 40–49.

[32] N. Zazworka, C. Izurieta, S. Wong, Y. Cai, C. Seaman, F. Shull *et al.*, "Comparing four approaches for technical debt identification," *Software Quality Journal*, Vol. 22, No. 3, 2014, pp. 403–426.

[33] X. Zhu, Q. Song, and Z. Sun, "Automated identification of change-prone classes in open source software projects." *Journal of Software*, Vol. 8, No. 2, 2013, pp. 361–366.

[34] M. Lindvall, "Are large C++ classes change--prone? An empirical investigation," *Software: Practice and Experience*, Vol. 28, No. 15, 1998, pp. 1551–1558.

[35] M. Lindvall, "Measurement of change: stable and change-prone constructs in a commercial C++ system," in *Proceedings Sixth International Software Metrics Symposium*. IEEE, 1999, pp. 40–49.

[36] Y. Liu and T.M. Khoshgoftaar, "Genetic programming model for software quality classification," in *Proceedings Sixth International Symposium on High Assurance Systems Engineering. Special Topic: Impact of Networking*. IEEE, 2001, pp. 127–136.

[37] M. Al-Khiaty, R. Abdel-Aal, and M.O. Elish, "Abductive network ensembles for improved prediction of future change-prone classes in object-oriented software." *International Arab Journal of Information Technology*, Vol. 14, No. 6, 2017, pp. 803–811.

[38] T.M. Khoshgoftaar, N. Seliya, and Y. Liu, "Genetic programming-based decision trees for software quality classification," in *15th International Conference on Tools with Artificial Intelligence*. IEEE, 2003, pp. 374–383.

[39] L. Kumar, S.K. Rath, and A. Sureka, "Empirical analysis on effectiveness of source code metrics for predicting change-proneness," in *10th Innovations in Software Engineering Conference*. ACM, 2017, pp. 4–14.

[40] N. Tsantalis, A. Chatzigeorgiou, and G. Stephanides, "Predicting the probability of change in object-oriented systems," *IEEE Transactions on Software Engineering*, Vol. 31, No. 7, 2005, pp. 601–614.

[41] L. Kumar, S.K. Rath, and A. Sureka, "Using source code metrics to predict change-prone web services: A case-study on ebay services," in *Workshop on Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE)*. IEEE, 2017, pp. 1–7.

[42] A.R. Sharafat and L. Tahvildari, "Change prediction in object-oriented software systems: A probabilistic approach," *Journal of Software*, Vol. 3, No. 5, 2008, pp. 26–39.

[43] L. Kumar, R.K. Behera, S. Rath, and A. Sureka, "Transfer learning for cross-project change-proneness prediction in object-oriented software systems: A feasibility analysis," *ACM SIGSOFT Software Engineering Notes*, Vol. 42, No. 3, 2017, pp. 1–11.

[44] D. Azar, "A genetic algorithm for improving accuracy of software quality predictive models: A search-based software engineering approach," *International Journal of Computational Intelligence and Applications*, Vol. 9, No. 02, 2010, pp. 125–136.

[45] R. Malhotra and R. Jangra, "Prediction and assessment of change prone classes using statistical and machine learning techniques," *Journal of Information Processing Systems*, Vol. 13, No. 4, 2017, pp. 778–804.

[46] A.R. Han, S.U. Jeon, D.H. Bae, and J.E. Hong, "Measuring behavioral dependency for improving change-proneness prediction in uml-based design models," *Journal of Systems and Software*, Vol. 83, No. 2, 2010, pp. 222–234.

[47] R. Malhotra and M. Khanna, "An empirical study for software change prediction using imbalanced data," *Empirical Software Engineering*, Vol. 22, No. 6, 2017, pp. 2806–2851.

[48] S. Eski and F. Buzluca, "An empirical study on object-oriented metrics and software evolution in order to reduce testing costs by predicting change-prone classes," in *Fourth International Conference on Software Testing, Verification and Validation Workshops*. IEEE, 2011, pp. 566–571.

[49] M. Yan, X. Zhang, C. Liu, L. Xu, M. Yang, and D. Yang, "Automated change-prone class prediction on unlabeled dataset using unsupervised method," *Information and Software Technology*, Vol. 92, 2017, pp. 1–16.

[50] A. Agrawal and R.K. Singh, "Empirical validation of OO metrics and machine learning algorithms for software change proneness prediction," in *Towards Extensible and Adaptable Methods in Computing*. Springer, 2018, pp. 69–84.

[51] C. Liu, Y. Dan, X. Xin, Y. Meng, and Z. Xiaohong, "Cross-project change-proneness prediction," in *42nd Annual Computer Software and Applications Conference (COMPSAC)*. IEEE, 2018, pp. 64–73.

[52] R. Malhotra and M. Khanna, "Investigation of relationship between object-oriented metrics and

change proneness," *International Journal of Machine Learning and Cybernetics*, Vol. 4, No. 4, 2013, pp. 273–286.

[53] L. Kaur and M. Ashutosh, "A comparative analysis of evolutionary algorithms for the prediction of software change," in *International Conference on Innovations in Information Technology (IIT)*. IEEE, 2018, pp. 188–192.

[54] R. Malhotra and A.J. Bansal, "Cross project change prediction using open source projects," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2014, pp. 201–207.

[55] R. Malhotra and M. Khanna, "Prediction of change prone classes using evolution-based and object-oriented metrics," *Journal of Intelligent and Robotic Systems Fuzzy Systems*, Vol. 34, No. 3, 2018, pp. 1755–1766.

[56] R. Malhotra and M. Khanna, "A new metric for predicting software change using gene expression programming," in *5th International Workshop on Emerging Trends in Software Metrics*. ACM, 2014, pp. 8–14.

[57] R. Malhotra and M. Khanna, "Particle swarm optimization-based ensemble learning for software change prediction," *Information and Software Technology*, Vol. 102, 2018, pp. 65–84.

[58] C. Marinescu, "How good is genetic programming at predicting changes and defects?" in *16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. IEEE, 2014, pp. 544–548.

[59] M.O. Elish, H. Aljamaan, and I. Ahmad, "Three empirical studies on predicting software maintainability using ensemble methods," *Soft Computing*, Vol. 19, No. 9, 2015, pp. 2511–2524.

[60] R. Malhotra and M. Khanna, "Mining the impact of object oriented metrics for change prediction using machine learning and search-based techniques," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2015, pp. 228–234.

[61] A. Bansal, "Empirical analysis of search based algorithms to identify change prone classes of open source software," *Computer Languages, Systems and Structures*, Vol. 47, 2017, pp. 211–231.

[62] S.R. Chidamber and C.F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on Software Engineering*, Vol. 20, No. 6, 1994, pp. 476–493.

[63] J. Bansiya and C.G. Davis, "A hierarchical model for object-oriented design quality assessment," *IEEE Transactions on Software Engineering*, Vol. 28, No. 1, 2002, pp. 4–17.

[64] M. Lorenz and J. Kidd, *Object-oriented software metrics: A practical guide*. Prentice-Hall, Inc., 1994.

[65] W. Li and S. Henry, "Object-oriented metrics that predict maintainability," *Journal of Systems and Software*, Vol. 23, No. 2, 1993, pp. 111–122.

[66] K. Gao, T.M. Khoshgoftaar, and A. Napolitano, "Combining feature subset selection and data sampling for coping with highly imbalanced software data," in *Software Engineering Knowledge Engineering Conference*, 2015, pp. 439–444.

[67] H. He and E.A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 21, No. 9, 2009, pp. 1263–1284.

[68] C.G. Weng and J. Poon, "A new evaluation measure for imbalanced datasets," in *7th Australian Data Mining Conference*. Australian Computer Society, Inc., 2008, pp. 27–32.

[69] M.A. De Almeida and S. Matwin, "Machine learning method for software quality model building," in *International symposium on methodologies for intelligent systems*. Springer, 1999, pp. 565–573.

[70] R. Malhotra, *Empirical research in software engineering: Concepts, analysis and applications*. CRC Press, 2016.

[71] W. Afzal and R. Torkar, "On the application of genetic programming for software engineering predictive modeling: A systematic review," *Expert Systems with Applications*, Vol. 38, No. 9, 2011, pp. 11 984–11 997.

[72] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," *Information and Software Technology*, Vol. 54, No. 1, 2012, pp. 41–59.

Appendix A.

Table A1 states the results of data extraction, i.e. the key parameters extracted from each primary study. Table A2 states the datasets that have been used by at least two primary studies.

Table A1. Key parameters of primary studies

S.No.	Year	Venue	Predictors	Datasets	Data Analysis algorithms	Performance measures	Validation method	Statistical test
PS1	2001	Conf.	NOCI, 4 Lines of code measures	Windows based software application (VLWA dataset)	LR, GP	Type I error, Type II error, Overall misclassification rate	Hold-out	–
PS2	2003	Conf.	NOCI, 4 Lines of Code measures	Windows based Software Application (VLWA Dataset)	GP, Decision tree based GP	Type I error, Type II error, Overall misclassification rate	Hold-out	–
PS3	2005	Journal	CK, NOM, Probability values related to history, Probability of Change	JFlex, JMol	LR	Accuracy, Recall, False Positive Ratio, False Negative Ratio, Goodness of Fit	–	–
PS4	2005	Journal	AID, ALD, CC, SLOC, NOP, MNOB, MPC, NOLV, NIC	JFlex	Non-linear system of equations, Linear system of equations, Depth first search graphs, Binary dependencies	Accuracy, False Positive Ratio, False Negative Ratio	–	–
PS5	2010	Journal	22 OO metrics: 4 cohesion metrics, 4 coupling metrics, 7 inheritance metrics, 7 size metrics	Bean Browser, Ejbvoyager, Free, Javamapper, Jchempaint, Jedit, Jetty, Jigsaw, Jlex, Lmjs, Voji, 4 versions of JDK	C4.5, GA	Accuracy, J-Index	Ten-fold	–
PS6	2010	Journal	CK, Lorenz and Kidd [64], MOOD metrics, BDM	Jflex (13 versions)	Stepwise Multiple Regression	Goodness of Fit	–	ANOVA
PS7	2011	Journal	22 OO metrics: 4 cohesion metrics, 4 coupling metrics, 7 inheritance metrics, 7 size metrics	Bean Browser, Ejbvoyager, Free, Javamapper, Jchempaint, Jedit, Jetty, Jigsaw, Jlex, Voji	C4.5, ACO	Accuracy	Ten-fold	Wilcoxon Signed Rank

Table A1 continued

S.No.	Year	Venue	Predictors	Datasets	Data analysis algorithms	Performance measures	Validation method	Statistical test
PS8	2011	Conf.	CK, QMOOD	Yari (3 versions), UCdetector (4 versions), JFreeChart (4 versions)	Combined Rank List Mechanism	Hit-Ratio (Recall), Change Cost, Cost ratio	–	–
PS9	2011	Journal	62 OO metrics: 18 cohesion metrics, 20 coupling metrics, 17 inheritance metrics, 7 size metrics	102 Java Systems	Random effect meta-analysis	AUC	–	–
PS10	2011	Technical Report	CK, 3 usage metrics, 3 complexity metrics, Interface Usage Cohesion metric	8 Eclipse plug-in projects and 2 Hibernate Systems	NB, SVM, MLP	Recall, Precision, AUC	Ten-Fold	Wilcoxon Signed Rank
PS11	2012	Conf.	CK, 7 Network centrality measures from social network analysis	19 Eclipse plug-in projects, Azureus	MLP, BN	Recall, Precision, AUC	Ten-Fold	Friedman, Wilcoxon Signed rank
PS12	2013	Journal	CK, 16 Evolution-based metrics	PeerSim (9 versions), VSSPlugin (13 versions)	LR	Accuracy	Ten-Fold	Wilcoxon Signed Rank
PS13	2013	Journal	CK, 16 other class-level metrics	Frinika, FreeMind, OrDrumBox	LR, RF, MLP, BG	Recall, Specificity, AUC	Ten-Fold	–
PS14	2014	Conf.	CK, SLOC	Apache Abdera (4 versions), Apache POI (4 versions), Apache Rave (4 versions)	LB	Precision, AUC	Ten-Fold, Cross-project	–
PS15	2014	Conf.	CK, SLOC	Simutrans, Glest	GEP	Accuracy, AUC	Ten-Fold	–
PS16	2014	Conf.	NOM, DIT, RFC, NOC, CBO, TCC, SLOC	ArgoUmL, Findbugs, FOP, FreeCol	GP	Recall, Precision	–	Proportion test
PS17	2015	Journal	CK	PeerSim, VSSPlugin	LR, MLP, RBF, SVM, DT, GEP, *k*-means, Ensemble of Models (Best in training, Bagging, Boosting, Majority Voting, Non-linear Decision tree Forest)	Accuracy, AUC	Hold-out, leave-one out	–

Table A1 continued

S.No.	Year	Venue	Predictors	Datasets	Data analysis algorithms	Performance measures	Validation method	Statistical test
PS18	2015	Conf.	CK, SLOC, NOM, NIV, NPM, NIM, NOA	Simutrans, Glest, Celestia	LR, RF, BG, MLP, AB, CPSO, HIDER, MPLCS, SUCS, GFS-SP, NNEP	Accuracy, AUC, Precision, Specificity, Recall, F-measure, G-measure	Ten-fold, Cross-project	Friedman
PS19	2017	Journal	CK, SLOC	Apache Rave, Apache Math	NB, BN, LB, AB, GFS-AB, GFS-LB, GFS-MLB, HIDER, NNEP, PSO-LDA, GFS-GP, GFS-SP, SLAVE	Accuracy, G-mean	Ten-fold	Wilcoxon Signed Rank
PS20	2017	Journal	CK, 16 Evolution-based metrics	VSSPlugin (13 versions)	GMDH	Accuracy, AUC, Precision, Recall, F-measure	Hold-out	–
PS21	2017	Conf.	62 OO metrics: 19 cohesion metrics, 19 coupling metrics, 17 inheritance metrics, 7 size metrics	Eclipse	LR, NB, Extreme Machine Learning (Linear, Polynomial and RBF kernels), SVM (Linear, Polynomial and Sigmoid kernels), Ensembles of Techniques (Best in Training, Majority Voting)	Accuracy, AUC	Ten-fold	–
PS22	2017	Conf.	21 OO metrics including CK metrics	Ebay Services (5 versions)	Least Square SVM (Linear, Polynomial and RBF kernels)	Accuracy, F-measure	Twenty-fold	*t*-test
PS23	2017	Journal	61 OO metrics	10 Eclipse plug-ins	LR, MLP, RBF, DT, RF, Ensembles of Techniques (Best in Training, Majority Voting, Non-Linear Decision Tree Forest)	Accuracy, Precision, Recall, F-measure	Ten-fold, Cross-project	Wilcoxon Signed Rank
PS24	2017	Journal	13 OO metrics including CK suite	AOI, SweetHome 3D	LR, RF, AB, BG, MLP, NB, BN, J48, NNGE	Recall, Specificity, AUC	Ten-fold, Cross-project	*t*-test

Table A1 continued

S.No.	Year	Venue	Predictors	Datasets	Data analysis algorithms	Performance measures	Validation method	Statistical test
PS25	2017	Journal	18 OO metrics including CK suite, SLOC, QMOOD suite, AC, EC, LCOM3, AMC, IC, CBM	Six Android application packages	PSO-LDA, NNEP, GFS-LB, CART, SUCS, CPSO, C4.5, GA-ADI, HIDER, MLP-CG, MPLCS, LDA, DT-GA, XCS, SVM	Recall, PF, Balance, G-mean	Ten-fold	Friedman, Wilcoxon Signed Rank
PS26	2017	Journal	18 OO metrics including CK suite, SLOC, QMOOD suite, AC, EC, LCOM3, AMC, IC, CBM	Three Android application packages, Net, IO, Log4j	MLP, RF, NB, AB, LB, BG	Recall, Precision, Accuracy, AUC, Balance, G-mean	Ten-fold, Inter-version	Friedman, Wilcoxon Signed Rank
PS27	2017	Journal	10 OO metrics including CK suite, Li and Henry [65], SIZE1	Ant, Antlr, Argouml, Azureus, Freecol, Freemind, Hibernate, Jgraph, Jmeter, Jstock, Jung, Junit, Lucene, Weka	LB, MLP, RBF, SVM, *k*-means, CLAMI, CLAMI+	Accuracy, AUC, F-measure	Within project, Cross-project	Friedman, Nemenyi
PS28	2018	Chapter	15 OO metrics including CK suite	GATE, Tuxguitar, FreeCol, KolMafia, Legatus	MLP, LR, RF, KStar, PART, BG, BN	Recall, Specificity, AUC, F-measure	Ten-fold	Friedman
PS29	2018	Journal	Entropy of changes, number of developers, structural and semantic scattering of developers, evolutio-based metrics, OO metrics	Ant, Cassandra, Lucene, POI, Synapse, Velocity, Xalan, Xerces, ArgoUML, aTunes, FreeMind, JEdit, JFreeChart, JHotDraw, JVLT, pBeans, pdfTranslator, Redaktor, Serapion, Zuzel	LR	Recall, Precision, AUC, MCC, Brier Score	3 month sliding window to train and test models	Mann–Whitney, Cliff
PS30	2018	Conf.	CK, SLOC	ArgoUML, FreeCol, JMeter, Jung, Weka (4 versions each)	LR, NB, DT, SVM, Decision Table, Deep Metric Learning	Recall, Precision, F-Measure	Cross-project	–

Table A1 continued

S.No.	Year	Venue	Predictors	Datasets	Data analysis algorithms	Performance measures	Validation method	Statistical test
PS31	2018	Conf.	CK	Ant, Antlr, ArgoUML, Azureus, FreeCol, FreeMind, Hibernate, JGraph, JMeter, JStock, Jung, JUnit, Lucene, Weka	BN	AUC	Cross-project	Wilcoxon Signed Rank
PS32	2018	Conf.	TCC, CHL, CHV, CHE, CHB, MI, AC, EC, Instability	JFreeChart (4 versions)	LR, LDA, GFS-GP, GFS-LB, GFS-SP, GFS-AB, NNEP, GANN	Accuracy	Ten-fold	Friedman, Wilcoxon Signed Rank
PS33	2018	Journal	CK, 16 Evolution-based metrics	Android Contacts (5 versions), Android Gallery2 (4 versions)	LR, MLP, NB, RF, AB, BG, LB	Accuracy, AUC	Ten-fold	Friedman, Wilcoxon Signed Rank
PS34	2018	Journal	CK, SLOC	Six Android application packages, IO, Net, Math, Log4j	RF, BG, AB, LB, 4 CPSO voting based fitness ensembles	Balance, G-Mean	Ten-fold	Friedman, Wilcoxon Signed Rank
PS35	2018	Journal	Complexity metrics, Word metrics, Network Metrics	Ant, Eclipse, JEdit, Itextpdf, Liferay, Lucene, Struts, Tomcat	C4.5, NB, SVM	Recall, Precision, F-measure, AUC, MCC	Ten-fold	Scott–Knott
PS36	2019	Journal	OO metrics, Process metrics, Developer related factors	Ant, Log4j, Lucene, Pbeans, POI, Synapse, Velocity, Xalan, Xerces, JEdit	LR, Simple Logistic, NB, MLP, AB, BG, RF, Voting	F-measure, AUC, MCC	Ten-fold	Scott–Knott, Cliff

Table A1 continued

S.No.	Year	Venue	Predictors	Datasets	Data analysis algorithms	Performance measures	Validation method	Statistical test
PS37	2019	Conf.	20 OO metrics including CK metrics suite	compare, webdav, debug, update, core, swt, team, pde, ui, jdt	LR, Linear Regression, Polynomial Regression, DT, SVM (Linear, Polynomial, RBF), Extreme ML (Linear, Polynomial, RBF), Least-Square SVM (Linear, Polynomial, RBF) Simple Logistic, Neural network with 5 training algorithms, Ensembles of Techniques (Best in Training, Majority Voting, Non-Linear Decision Tree Forest)	Accuracy, F-measure	Five-fold	Wilcoxon Signed Rank
PS38	2019	Journal	CK, SLOC	AOI, CLick, DrJava, Giraph, Gora, Hama, HyperSQL DB, JabRef, JMeter, JEdit, LogicalDoc, Maven, Phoenix, SubSonic, ZooKeeper	LR, AB, BG, RF, LB, 4 CPSO voting based fitness ensembles, ASOF Classifier	F-measure, AUC, MCC	Ten-fold	Scott–Knott, Cliff

Note: "–" indicates the corresponding information was not found in the study.

AC: Afferent Coupling; AID: Access of Imported Data; ALD: Access of Local Data; AMC: Average Method Complexity; ASOF: Adaptive Selection of Optimum Fitness; BDM: Behavioral Dependency Measurement; CBM: Coupling Between Methods of a Class; Conf.: Conference; CC: Cyclomatic Complexity; CLAMI: Clustering Lableing Metric selection and Instance selection; EC: Efferent Coupling; IC: Inheritance Coupling; MCC: Mathews Correlation Coefficient; MNOB: Maximum Number Of Branches; MOOD: Metrics for Object-Oriented Design; MPC: Message Passing Coupling; NIC: Number of Imported Classes, NIM: Number of Instance Methods; NIV: Number of Instance Variables; NOCI: Number of Times Source File was Inspected; NOLV: Number Of Local Variables; NOA: Number of Attributes; NOM: Number of Methods per Class; NOP: Number Of Parameters; NPM: Number of Public Methods; RBF: Radial Basis Function; TCC: Total Cyclomatic Complexity; CHL: Cumulative Halstead Length; CHV: Cumulative Halstead Volume; CHE: Cumulative Halstead Effort; CHB: Cumulative Halstead Bugs; MI: Maintainability Index; QMOOD: Quality Model for Object-Oriented Design.

Table A2. Commonly used datasets

Dataset Name	Study Numbers
Android Bluetooth	PS27, PS26, PS34
Android Calendar	PS27, PS26, PS34
Android Contacts	PS26, PS33, PS34
Android Gallery	PS26, PS33, PS34
Android MMS	PS27, PS26, PS34
Android Telephony	PS26, PS34
Ant	PS27, PS29, PS31, PS35, PS36
Antlr	PS27, PS31
AOI	PS24, PS38
ArgoUML	PS16, PS27, PS29, PS30, PS31
Azureus	PS11, PS27, PS31
Bean Browser	PS5, PS7
Eclipse	PS10, PS11, PS21, PS23, PS35
Free	PS5, PS7
FreeCol	PS16, PS27, PS28, PS30, PS31
FreeMind	PS13, PS27, PS29, PS31
Glest	PS15, PS18
Hibernate	PS10, PS27, PS31
IO	PS27, PS34
JChempaint	PS5, PS7
JEdit	PS5, PS29, PS35, PS36, PS38
Jetty	PS5, PS7
JFlex	PS3, PS4
JFreeChart	PS8, PS29, PS32
JGraph	PS27, PS31
Jigsaw	PS5, PS7
Jlex	PS5, PS7
JavaMapper	PS5, PS7
JMeter	PS30, PS31, PS38
Jung	PS30, PS31
Log4j	PS27, PS34, PS36
Lucene	PS29, PS31, PS35, PS36
Math	PS19, PS34
Net	PS27, PS34
PeerSim	PS12, PS17
POI	PS14, PS29, PS36
Synapse	PS29, PS36
Velocity	PS29, PS36
Voji	PS5, PS7
VSSPlugin	PS12, PS17, PS20
Weka	PS30, PS31
Windows based software application (VLWA)	PS1, PS2
Xalan	PS29, PS36

e-Informatica Software Engineering Journal (eISEJ) is an international, open access, no authorship fees, blind peer-reviewed journal that concerns theoretical and practical issues pertaining development of software systems. Our aim is to focus on experimentation and machine learning in software engineering.

The journal is published under the auspices of the Software Engineering Section of the Committee on Informatics of the Polish Academy of Sciences and Wrocław University of Science and Technology.

Aims and Scope:

The purpose of **e-Informatica Software Engineering Journal** is to publish original and significant results in all areas of software engineering research.

The scope of **e-Informatica Software Engineering Journal** includes methodologies, practices, architectures, technologies and tools used in processes along the software development lifecycle, but particular stress is laid on empirical evaluation.

e-Informatica Software Engineering Journal is published online and in hard copy form. The on-line version is from the beginning published as a gratis, no authorship fees, open access journal, which means it is available at no charge to the public. The printed version of the journal is the primary (reference) one.

Topics of interest include, but are not restricted to:

— Software requirements engineering and modeling
— Software architectures and design
— Software components and reuse
— Software testing, analysis and verification
— Agile software development methodologies and practices
— Model driven development
— Software quality
— Software measurement and metrics
— Reverse engineering and software maintenance
— Empirical and experimental studies in software engineering (incl. replications)
— Evidence based software engineering
— Systematic reviews and mapping studies
— Meta-analyses
— Object-oriented software development
— Aspect-oriented software development
— Software tools, containers, frameworks and development environments
— Formal methods in software engineering.
— Internet software systems development
— Dependability of software systems
— Human-computer interaction
— AI and knowledge based software engineering
— Data mining in software engineering
— Prediction models in software engineering
— Mining software repositories
— Search-based software engineering
— Multiobjective evolutionary algorithms
— Tools for software researchers or practitioners
— Project management
— Software products and process improvement and measurement programs
— Process maturity models

Important information: Papers can be rejected administratively without undergoing review for a variety reasons, such as being out of scope, being badly presented to such an extent as to prevent review, missing some fundamental components of research such as the articulation of a research problem, a clear statement of the contribution and research methods via a **structured abstract** or the evaluation of the proposed solution (empirical evaluation is strongly suggested).

Funding acknowledgements: Authors are requested to identify who provided financial support for the conduct of the research and/or preparation of the article and to briefly describe the role of the sponsor(s), if any, in study design; in the collection, analysis and interpretation of data; in the writing of the paper. If the funding source(s) had no such involvement then this should be stated as well.

The submissions will be accepted for publication on the base of positive reviews done by international Editorial Board (`http://www.e-informatyka.pl/index.php/einformatica/editorial-board/`) and external reviewers. English is the only accepted publication language. To submit an article please enter our online paper submission site (`https://mc.manuscriptcentral.com/e-InformaticaSEJ`).

Subsequent issues of the journal will appear continuously according to the reviewed and accepted submissions.

http://www.e-informatyka.pl/wiki/e-Informatica

e-Informatica

ISSN 1897-7979