Capturing Cost Avoidance through Reuse: Systematic Literature Review and Industrial Evaluation

Mohsin Irshad
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden
mohsin.irshad@bth.se

Richard Torkar
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
richard.torkar@gmail.com

Kai Petersen
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden
kai.petersen@bth.se

Wasif Afzal
Malardalen University
Box 883, 721 23 Västeras Sweden
wasif.afzal@mdh.se

ABSTRACT

Background: Cost avoidance through reuse shows the benefits gained by the software organisations when reusing an artefact. Cost avoidance captures benefits that are not captured by cost savings e.g. spending that would have increased in the absence of the cost avoidance activity. This type of benefit can be combined with quality aspects of the product e.g. costs avoided because of defect prevention. Cost avoidance is a key driver for software reuse. **Objectives:** The main objectives of this study are: (1) To assess the status of capturing cost avoidance through reuse in the academia; (2) Based on the first objective, propose improvements in capturing of reuse cost avoidance, integrate these into an instrument, and evaluate the instrument in the software industry. **Method:** The study starts with a systematic literature review (SLR) on capturing of cost avoidance through reuse. Later, a solution is proposed and evaluated in the industry to address the shortcomings identified during the systematic literature review. **Results:** The results of a systematic literature review describe three previous studies on reuse cost avoidance and show that no solution, to capture reuse cost avoidance, was validated in industry. Afterwards, an instrument and a data collection form are proposed that can be used to capture the cost avoided by reusing any type of reuse artefact. The instrument and data collection form (describing guidelines) were demonstrated to a focus group, as part of static evaluation. Based on the feedback, the instrument was updated and evaluated in industry at 6 development sites, in 3 different countries, covering 24 projects in total. **Conclusion:** The proposed solution performed well in industrial evaluation. With this solution, practitioners were able to do calculations for reuse costs avoidance and use the results as decision support for identifying potential artefacts to reuse.

EASE 2016 Limerick, Ireland

DOI: 10.475/123_4

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous Measurements, Cost Avoidance, Cost Savings

Keywords

Cost Avoidance, Cost Savings, Software Reuse

1. INTRODUCTION

Software reuse helps the organisations in decreasing costs while still meeting certain quality requirements [1] [2]. However, introducing a reuse culture in an organisation is difficult because software reuse is concerned with various aspects of software development [3]. Technical, management and business aspects of software reuse are still being researched [4] [5]. The measurements related to software reuse have been reported in various studies [6] [7]. A study by Frakes [8] described different types of reuse metrics such as costs related to software reuse, levels of software reuse, measuring reusability, etc. These measurements provide a foundation for building, maintaining and improving the reuse processes and reusable products.

As the practitioners develop a software product they come across problems that they have solved in the past. In these cases, they often reuse the solution they had developed in the past or they reuse a known solution from someone else [9]. This type of reuse, which is not planned in advance, is known as pragmatic (ad-hoc) reuse [10] [11]. Studies have shown that a considerable reuse takes place in the form of pragmatic reuse [9] [12]. This type of reuse takes place in a white-box way (needs modification before reusing) [13]. The effort spent on making this component reusable and costs avoided (in the form of money or effort) because of this reuse instance are hard to know in advance. The systematic reuse approaches, such as product lines [14], have been proposed, but ad-hoc reuse takes place on a frequent manner [9] [12].

A variety of software reuses cost models already exist in literature [15] [16]. Poulin [16] categorized these cost models into three types: Return on Investment models, Cost Benefit Analysis models and Cost Avoidance (CA) models. These models provide solution to different problems. Return on Investment models measure benefits for organisations that have already invested in reuse processes. Cost-Benefit models are used for making decisions on reuse investments and these are useful for management in making optimal decisions about software reuse. Cost avoidance models help to calculate the costs avoided by reusing an artefact. These have a limited scope as these are only applied after the artefact has been reused and these models (CA) do not impose any restriction on reuse practices or investments.

In the context of this study we have followed the definition of cost avoidance provided by Poulin, who defines cost avoidance as the financial benefits in terms of money that the organisation has prevented to keep from occurring [16]. This prevention can take place because of ad-hoc reuse or improvement in quality of software, thus causing fewer bugs or buying third party product at lesser cost than quoted in contract. An example of cost avoidance is provided in Section 2.

The cost avoidance through reuse is difficult to measure because of two reasons: (i) Cost avoidance usually takes place in pragmatic software reuse (also known as ad-hoc reuse) and it is hard to predict how much costs can be avoided. In the systematic reuse process or in software product lines the reusable artefacts can be already identified, their modification costs can be estimated and these reuse instances can be planned by the organisations. (ii). Reuse artefacts, as presented in literature, are of various different types [17]. With the introduction of various types of reuse artefacts, measuring cost avoidance through reuse can become difficult. There are no agreed upon guidelines that can help organisations in measuring reuse cost avoidance. This study attempts to address these issues. The contributions of this study are:

- **Contribution 1:** To investigate the reuse *cost avoidance* measurement solutions present in the literature.
- **Contribution 2:** To identify an *artefact-independent* method that could help the organisations to measure reuse cost avoidance (CA).
- **Contribution 3:** To evaluate the identified *artefact-independent* method, in the industry, for measuring cost avoidance (CA) through reuse.

It is worth mentioning that the study is not aimed at understanding general reuse culture or processes, it is limited to cost avoidance measurements only.

Section 2 describes the different aspects of cost avoidance. In the first phase (Section 3) of the study we investigate the state of the art in relation to reuse cost avoidance. This is done with a systematic literature review on software reuse cost avoidance. The recommendations from the first phase are presented in Section 4. Based on the recommendations, a cost avoidance measurement instrument is proposed in Section 5 . The instrument is evaluated dynamically by its application to 24 industrial projects, described in Section 6. Section 7 discusses the results and the approach used for calculating reuse cost avoidance. Threats to the validity of the study are described in Section 8. Section 9 concludes the paper. An overview of the complete study is shown in Fig. 1.

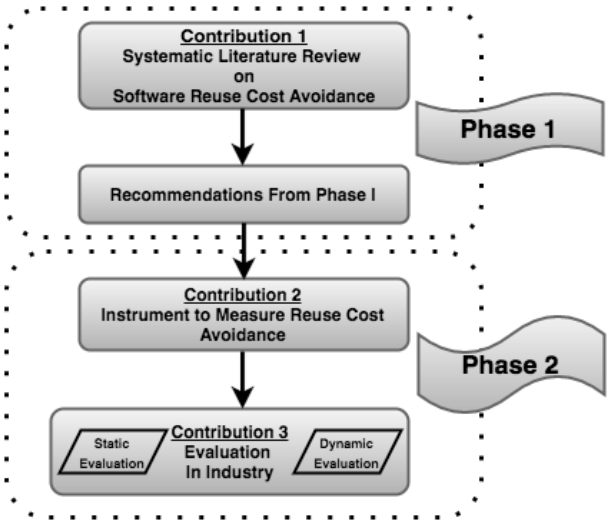

Figure 1: The various steps in this study.

2. BACKGROUND:

We provide the definitions important in the context of this study, and thereafter provide an example of cost avoidance through reuse and make the difference to cost savings explicit.

2.1 Definition of terms

The terms "cost savings", "cost avoidance" and "return on investment (ROI)" are describing different types of benefits. The "cost savings" are the benefits that take place when an action *reduces* the costs. These savings are usually planned and can be measured more easily. "Cost avoidance" takes place when an action *prevents* (avoids) the future costs to happen. These are the financial benefits gained without expending resources [18]. The "return on investment (ROI)" describes the ratio of financial benefits (e.g. *profits*) achieved from an investment. The cost avoidance benefits are often difficult to measure because of the ad-hoc nature of software reuse.

Cost avoidance is frequently discussed in scientific literature [19] [20] [21]. However, a variety of different definitions of cost avoidance exist in the literature. Laszlo described cost avoidance, in context of total quality management (TQM) as the costs avoided because of improvements in the quality of the product [22]. In a study on industry-university collaboration, Denis et al. described cost avoidance as the costs that did not incur because of the industry university collaboration [23]. Ashenbaum differentiates between "cost savings" and "cost avoidance" in a study on supply chain management [18]. According him, if there is an increase in output without any additional resource expenditures, then the amount that would have been spend to reach this increased output is the cost avoided. He describes an example where purchase price is lower than the "quoted" price of a product in the vendor contract, then this type of savings (quoted price - purchased price) are known as costs avoided.

In software engineering, Jacobson et al. describes reuse cost avoidance as a metric that reflects the reduction in the products development cost because of reuse [24]. According to Frakes, reuse cost avoidance is the reduced cost of a product because of software reuse [8]. Guerrieri, et al. described cost avoidance as the costs that are avoided each time a reusable component is reused [25]. Lim differentiated between the cost avoidance from cost savings in an example in his study on software reuse [26]. According to him the costs avoided through reuse are the type of savings, in terms of money, that organisations were suppose to spend, but did not have to spend because of reusing an artefact.

2.2 Example of Cost Avoidance through reuse

Consider a hypothetical software organisation selling solutions to multiple customers. Assume that in the year 2012, the organisation spent $63,000 on the development of a customer project. Next year, in 2013, another customer project was completed that reused components from the previous project (from 2012) but needed $60,000. See the Table 1 for detail.

Table 1: Development Costs Per Year.

Year	Lines of Code	Cost / line of code	Total Costs
2012	90,000	$ 0.70	$ 63,000
2013	80, 000	$ 0.75	$ 60,000

Here a project manager can claim that savings were $ 3,000 ($ 63,000 - $ 60,000) when the organisation reused the components from the previous project. However, the project manager discovers that second project uses 10, 000 lines of code fewer than in the first project. So without reuse the organisation would have to write 10,000 additional LOCs. Due to increase in the salary of developers and taxes the development cost per line of code was also increased.
Cost without Reuse = cost per line of code (in $) X Lines of Code (LOC)

$$Cost\ without\ Reuse = \$\ 0.75 \times 90,000\ LOC = \$\ 67,500 \quad (1)$$

Therefore,

$$Costs\ Avoided\ = \$\ 67,500\ - \$\ 60,000 = \$\ 7,500 \quad (2)$$

The project manager can see that cost savings on the second project were $ 3,500, but cost avoidance through reuse was $ 7,500. The absence of reuse would have decrease the cost savings and cost avoidance. Therefore, the cost savings and cost avoidance describe different types of benefits to the user.

3. PHASE I: SYSTEMATIC LITERATURE REVIEW

In Phase I, a systematic literature review (SLR) was performed to gain understanding of state of the art with respect to measurement of cost avoidance through reuse. Kitchenham and Charters describe a number of steps for performing an SLR in [27], where they also list a number of benefits of conducting an SLR. The study was carried out based on the guidelines proposed by Kitchenham and Charters [27]. In the following sections we detail the research questions, data collection and results obtained from SLR.

3.1 Research Questions for SLR

In order to identify solutions for the measurement of cost avoidance through reuse (as present in literature [8] [24]), the following research questions were posed:

- **RQ 1:** Which studies describe a solution to calculate the cost avoided through reuse in a direct way?
- **RQ 2:** What metrics are required in order to apply each of these models?
- **RQ 3:** What are the limitations of the models?
- **RQ 4:** How have the models been evaluated?

3.2 Search Strategy

To construct the search terms used for the study the following steps were performed:

1. Develop and list the search terms based on the research questions.
2. From the list of terms, the synonyms of terms were added in the list of search terms.
3. New search terms were collected by changing the plurals to singular forms and singular to plural forms.
4. New search terms were collected by gathering keywords from abstracts and conclusions from a sample of relevant research papers.
5. New search terms were collected by browsing through grey literature (technical reports, non peer reviewed articles, and webpages)
6. New search terms were constructed by using Boolean `OR` with synonyms of search keywords and by using Boolean `AND` for combining different search terms
7. The search terms were piloted on the IEEE Xplore database. This piloting of the search strings helped in improving the quality of search terms.

All search terms, as used with different combinations, are listed in Table 2.

Different databases were selected to identify the relevant studies based on the guidelines proposed by Kitchenham and Charters [27]. The databases are: ACM Digital Library,IEEE Xplore, SpringerLink, ScienceDirect, Engineering Village, Wiley InterScience, ISI Web of Science.

Dybå and Dingsøyr's procedure [28] was followed for managing citations during this study. Endnote was used for storing the citations and removing any duplicate studies in the initial phase. During later stages of the study, spreadsheets were maintained for the each subsequent steps.

3.2.1 Study Selection Criteria

Keeping in mind the research objectives of this study, 'Cost Avoidance through reuse' [16] was selected as the main focus. Study selection criteria were divided into exclusion criteria and inclusion criteria.

Inclusion Criteria

- Discusses cost related aspects of software reuse with a focus on calculating cost avoidance.
- Should be published in a journal, conference or workshop (not necessarily peer reviewed!)
- Describes a validation of the presented cost avoidance model, i.e an empirical study.

- Any type of methodology used in the study is valid, e.g. literature review, systematic review, industrial experience, case study or an experience report on reuse cost estimation models.

Exclusion Criteria Any study not related to Software reuse and cost avoidance was removed. The studies that were in a language other than English were removed. If the full text of the study was not available then the study was not included.

3.3 Data Collection

A data collection form was developed for extracting the information from the selected studies. The data extraction form capturing general information (article title, authors, references), context information (industry or academia, study of professionals or students), and information related to the measurement instrument and its evaluation (such as metrics, outcome of evaluations, and so forth). The number of papers excluded at different stages of the inclusion and exclusion process are shown in Fig. 2. After the first application of the inclusion and exclusion criteria a snowball sampling step has been performed (backward snowballing) [30] to determine potentially relevant articles from the reference lists. The objective of this review was to find studies that described the measurement of costs avoided through reuse in software development. Therefore, these 30 papers were manually reviewed in-depth. While checking the complete paper it became apparent that the focus was not on cost avoidance, hence finally only three models were found that directly addressed the costs avoided through software reuse. These three primary studies were presenting measure for capturing cost avoidance [33, 16, 34].

Table 2: Results after initial search and search strings.

Search term	#papers
Software Reuse Cost Model	1545
Reuse Cost Model	3069
Software Reuse Cost Measurements	327
Software Reuse Cost Measuring Framework	172
Software Reuse Cost Estimation	567
Software Reuse Savings	678
Measuring Software Reuse Benefits	147
Software Reuse Economics	470
Reuse Economics	1414
Software Reuse Benefits	2326
Reuse Benefits	2560
Software Reuse Cost Avoidance	130
Reuse Cost Avoidance	400
Reuse Return On Investment	572
Software Reuse Return On Investment	422
Reuse Finance	180
Reuse Cost Estimation	927
Reuse Cost Assessment	843
Software Reuse Cost Estimation	369
Software Reuse Cost Assessment	189
Software Reuse Business	1777
Reusability Cost	1989
Software Reuse Cost Measurement Method	147
Software Reuse Cost Measurement Technique	118
Software Reuse Cost Reduction	490
Software Reuse Price	119
Tracking Reuse Cost	196
Monitoring Reuse Cost	413
Reuse Cost Benefit	1249
Software Reuse ROI	972
Software Reuse Financial Aspect	43
Total	24820

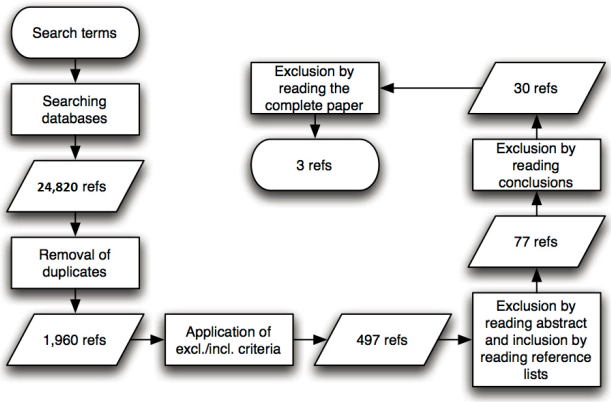

Figure 2: Multi-step filtering of studies and final number of primary studies.

These three articles had one thing in common; they provided a mechanism to measure the costs avoided through reuse. In addition, there were two other papers that could be seen as primary studies, [31] and [32], but these papers were very neatly summed up in [16] and, hence, we focused on the latest reference in this case.

Since only three primary studies were identified therefore quality assessment was done in a simple way. The details of quality assessment are not described in this study and these are available on web [29]. The other details obtained from each step of SLR are also provided on the web [29].

3.4 Results from SLR

3.4.1 *Models to capture cost avoidance (RQ1)*

In the following paragraphs a brief analyses of the three primary studies[16, 33, 34] are provided.

***Poulin's et al. model*:** The model was originally proposed in a study [31] and further described by Poulin in [16]. The model is applicable on individual projects or individual teams and is divided into two 'phases'. One phase is concerned with costs avoided during development and the other phase focuses on costs avoided during maintenance. According to this model, the direct cost avoided (DCA) can be estimated by the following equation:

$$DCA = RSI \times (1 - RCR) \times NCC \quad (3)$$

where RSI includes completely unmodified reused software lines of code, RCR is the relative cost of reuse (they use a factor of 0.2 i.e. cost of reuse) and, finally, NCC is new code costs. If the costs to develop a new component equals one unit of effort then the Relative Cost of Reuse (RCR) can be defined as "portion of this effort that is required to reuse a similar component without modification

(black-box)." Here without modification means searching, identifying, integrating and testing the reusable artefact [16]. Additionally, Poulin et al. calculate the service cost avoidance (SCA) of the reused software as:

$$SCA = RSI \times ER \times EC \tag{4}$$

where ER is the rate of errors reported during maintenance/service and EC is the cost incurred in reporting errors and retesting fixes. Poulin et al.'s final equation for reuse cost avoidance equals:

$$RCA = DSA + SCA \tag{5}$$

This model assumes that the users are aware of the costs per lines of code. The model is based on lines of code; therefore, no other software artefact can be measured with this model and, hence, points at a weakness with the model.

***Defense Information Systems Agency*:** United States Defense Information Systems Agency has proposed a model [33] similar to Poulin's et al. model [16]; in fact [33] could be seen as a more complex variant of that model. The following equation is used as a basis for this model:

$$Cost\ Avoided = R \times (En - Er) - COTS \tag{6}$$

where R is the costs of new custom code (in personnel hours). En is the estimated effort without reuse, and calculated by $a \times L$, where a is the relative cost of writing new code in personnel hours/lines of code, and L is the number of lines of new custom code, default for En is set to 1. Er is the estimated effort with reuse, also calculated by $a \times L$, however, a, the relative cost of verbatim reuse, defaults to 2, and L is the lines of verbatim reuse code. Finally, $COTS$ is the direct cost of buying the component (no mention of the indirect costs, e.g. learning how to use the component, can be found in [33]).

Unfortunately, this model has similar limitations as the previous model since it is based on, among other things, the number of lines of code and does not take into consideration any other forms of software reuse.

***Amar and Coffey's Model*:** Amar and Coffey provide three different models for measuring reuse savings [34]. They take into consideration the processes present in an organisation (ad hoc or systematic) and based on the processes one of the proposed formulas is used. The simplified formula for ad hoc reuse is thus:

$$\%\ \text{savings} = \left[SR - (TLR + U) \times \frac{\frac{N}{i}}{BUILD} - SR \times \frac{MOD}{BUILD} \right] \times 100 \tag{7}$$

where SR is the search hit rate (percentage of i that yielded a positive search result), TLR is the time to locate each potentially reusable item, U is the time to understand the suitability of each potentially reusable item (for the current task), N is the number of items that were examined, including each of the items that finally was reused, i is the number of attempted instances of reuse, MOD is the time to integrate/modify the reused item for current purposes and, finally, $BUILD$ is the time to build an element from scratch.

First, here they assume that organisations collect metrics such as search hit rate, time to understand suitability, etc. Secondly, this method provides a value regarding search hit rate equaling to 20%, though the reason behind this decision has not been made explicit. The above critique, limits the model's application, as most organisations are not interested in collecting complex metrics as it can be considered an overhead for the organisation [35]. This model also lacked clarity and necessary details such as how to collect data and what should be counted as reuse. However, it is clear, that the focus of the article was to provide a model for CA and validating the model in industry is not discussed (as required by the inclusion criteria).

3.4.2 Metrics Required (RQ2)

Two models were based on the number of lines of code [16, 33]. In order to apply these models, one has to know the number of lines of codes reused and the total size of the product in terms of lines of code. The third model, [34], was mainly based on personnel hours as a basic metric; however, it also included a number of corresponding metrics based on several other data points, e.g. search hit rate and number of attempted instances of reuse.

3.4.3 Assumptions of the Models (RQ3)

Reuse artefacts, as presented in literature, are of various different types [17]. The first thing to consider regarding the limitation of any model is to investigate the types of artefacts it can measure. Table 3 shows the artefacts each model can measure [16, 33, 34]. In some cases it was difficult to know whether the artefact could be measured or not, and in such cases '$\sim$' is used as an indicator.

3.4.4 Evaluation (RQ4)

By empirically evaluating in industry researchers can find out the limitations and advantages of their suggested approaches. The three primary studies were checked with respect to evaluation performed in industry. Both,Poulins [16] and DoDs [33], have been evaluated in industry (IBM and US Department of Defense, respectively), however details of the evaluations were missing. We could not find any indications that [34] contained a description of an evaluation in industry.

3.5 Conclusions from SLR

Table 3: Artefacts measured by models [17]. ($\surd$ indicates measures , $\sim$ indicates uncertainty, $\odot$ indicates do not measure)

Artefacts	[16]	[33]	[34]
Algorithms	$\odot$	$\odot$	$\sim$
Architectures	$\odot$	$\odot$	$\surd$
Data	$\odot$	$\odot$	$\surd$
Design	$\odot$	$\odot$	$\surd$
Documentation	$\odot$	$\odot$	$\surd$
Estimates	$\odot$	$\odot$	$\odot$
Human interfaces	$\odot$	$\odot$	$\surd$
Knowledge [36]	$\odot$	$\odot$	$\sim$
Models	$\odot$	$\odot$	$\surd$
Plans	$\odot$	$\odot$	$\surd$
Requirements	$\odot$	$\odot$	$\surd$
Service contracts	$\odot$	$\odot$	$\surd$
Test cases	$\odot$	$\odot$	$\surd$
Code	$\surd$	$\surd$	$\surd$

In previous sections an analysis on the various findings from the SLR was presented and, hence, the following conclusions were drawn accordingly:

- Lines of code, as a metric for calculating CA, cannot measure different types of reuse artefacts. Bhragv et al. have identified at least 14 different artefacts that can be reused therefore lines of code is not a relevant metric for many reusable artefacts [17]. Table 3 lists different reusable artefacts.
- The basic formula, for measuring the costs avoided through software reuse remains the same in all models, i.e. Costs without reuse - Costs with reuse. All models calculate reuse cost avoidance by the same basic formula; however, the metrics used in the formula differs between models. Two models used lines of code as a basic measure while one model used personnel hours as a basic measure for calculating software reuse. No temporal variable of costs was included, which is usually the case with cost-benefit reuse models (except for Poulin et al.'s model [31] that addresses net present value).
- No guidelines on what metrics to collect, for calculating cost avoidance through reuse, and how to collect metrics, related to a specific model, were found. The collection of metrics can cause problems for industry practitioners e.g. problems in using function sizes [38] and different definitions of lines of code [39] are well documented in the literature. None of the models provided guidelines.
- Only a limited number of studies is present on the industrial evaluation of models capable of calculation cost avoidance through reuse. One primary study was not evaluated in industry yet [34]. Literature contains evaluation experiences of two models only ([16, 33]). In an engineering discipline, it is important to evaluate the solutions in the industrial settings [40]. Industrial evaluation of studies acts as template and guidelines for other organisations that want to address similar problems. In particular, other aspects such as the origin of the reuse, e.g. COTS vs. Open source development, play a role, as well as the product where the reuse was taking place, and the point of time in which the reuse took place. Thus, these aspects need to be captured as well. This allows to retrospectively reflect and analyse the reuse practices in an organisation. In short, the reuse context has to be understood and captured, as context is generally important when interpreting data [41] [42].
- No limitations related to the models were reported. Negative findings help practitioners in evaluating a model before its usage and, hence, it can be seen as a major drawback associated with these models.

4. RECOMMENDATIONS FROM PHASE I

The research gaps identified (in Section 3.5) were transformed into recommendations for the next phase of the study. The recommendations are described below.

- Propose a metric that can calculate cost avoidance of different types of reusable software artefacts not dependent on measuring lines of code. Lines of code are not relevant for many artefacts that are reused in industry.
- Provide guidelines on the selection and collection of metrics required to calculate cost avoidance through reuse and the possibility to contextualise the collected metrics. There are no guidelines present in literature that can help organisations in measuring the cost avoidance through reuse and how to collect metrics required for this measurements.
- Provide a simple and intuitive instrument fulfilling the goal of understanding past cost avoidance as input for decision making for future cost avoidance. Simplicity here refers to only collecting a limited number of measures needed to achieve the goal.
- Perform industrial evaluation of measurements for cost avoidance through reuse. Only two articles were found in literature that mention industrial validation of models, though the details of validation were missing. Both these articles ([16, 33]) were based on lines of code.

The solution proposed in the Phase II (in Section 5 and Section 6) tries to incorporate these recommendations.

5. PHASE II - PROPOSED INSTRUMENT

The next section proposes (i) an instrument that is applicable to all types of reusable artefacts and the solution also provides (ii) guidelines on how to collect the required metrics, used in the instrument.

5.1 Selection of Metrics

In a study on industry relevant research, Kanso et al. identified "Simplicity" and "Usability" of a solution as vital aspects to consider when developing solution for the industry [43]. The successful technology transfer relies on the solution's simplicity and ease of use. Gencel et al. claimed that organisations have limited budget and resources for the collection of metrics [44]. It is important to select the metrics that are simple and easy to use. These two key inputs, (i) simplicity of solution and (ii) lack of resources for metric collection, were considered during the development of the instrument.

Studies have indicated the importance of reuse metrics in order to drive the reuse goals [45], and one of the few measures, common amongst most organisations, is 'personnel hours'. Many organisations collect personnel hours spent on tasks performed, as this measure is simple and easy to collect, hence, we believe that an instrument should use that metric as a foundation. Using personnel hours would also allow us to develop a more abstract method that could be used to measure costs avoided on every type of artefact, e.g. reuse of test cases or development environments (a development environment can consist of e.g. development kits such as frameworks, application servers, databases and build scripts). An added benefit of using this metric is the ease in which we can collect it and that no major guidelines are required for collecting it. A similar approach was used by [34] in their study.

5.2 The Instrument

The basic inspiration of the formula was taken up from Poulin's work [16]. However, a fundamental difference between his and our proposal is that instead of dealing with

lines of code, the new formula deals with personnel hours spent on reuse artefacts. Hence, we claim that the cost avoidance through reusing any software artefact can simply be measured using:

$$Cost\ Avoided = (O - I) \times H \quad (8)$$

Where O is the personnel hours spent on any artefact developed in-house, I is the personnel hours spent on, e.g. identifying, understanding and integrating an artefact to be reused and, finally, H is the cost (in any currency) of one personnel hour of work.

In many cases historical data is available if an artefact is an internal product of the organisation. This data can be used to establish personnel hours spent on an artefact when developed in-house, i.e. the factor O. The factor I on the other hand includes the time spent on, e.g. searching, understanding, integrating and testing the artefact, and the sum of all these phases is used as time spent on reusing the artefact. In a case, where historical data is not available an expert opinion can be utilised. Studies have shown that the expert opinion is a reliable method in software development when data is missing [46] [47] [48].

5.3 Solution

As a part of solution, a worksheet was developed that was generic enough to accommodate any organisation without needing to make any changes in it. This worksheet was then used to collect data and understand the context of reuse. The context of the reuse helps the practitioners in understanding the applicability of the reused artefact for future reuse opportunities. Table 4, shows the basic data fields that were collected.

This instrument should be used after the reuse. During the initial phases we do not know how much time (person-hours) would be spent on identifying an artefact (in pragmatic reuse [10]), changing it to fit our needs and product architecture and time required for its verification.

6. EVALUATION IN INDUSTRY

The evaluation was done in two steps, first conducting a static evaluation (collecting feedback from the practitioners), followed by a dynamic evaluation (application of the instrument to reflect on the reuse practices of the organisation), as suggested by Gorschek et al. [40].

Both dynamic and static validation were carried out at a large telecommunication company on twenty four different projects, in six globally distributed sites.

6.1 Static evaluation

The static validation was influenced by the method devised by Gorschek et al. keeping in view the industrial relevance of this study [40] . The purpose of static validation was to get the feedback from the practitioners, with the key question being how the instrument should be improved before it is taken into live operation.

A presentation and demonstration, of the instrument and guidelines for data collection, was made for a focus group having three designers, one tester and two managers. The solution was demonstrated with help of example use cases. The findings were shared with the user. Feedback was recorded in notes during the meeting.

6.1.1 Results from the Static Evaluation:

Several participants, of the focus group, objected on the formula used for calculating cost avoidance. They thought that the formula used should only be based on the time reported on a task and that it should not be based on figures present in literature since they believed that, for example, generalised values on the cost of reusing an artefact might differ quite radically from case to case. Therefore, contextualisation was important aspect to assess the similarities and differences between reuse cases, which resulted in the proposal of the data collection form shown in Table 4.

Another feedback was to extend the tool in order to evaluate the results from several different perspectives. Various types of charts, graphs and tables were added in order to analyse the results achieved by reusing artefacts.

Based on the above feedback, the instrument was updated to its final form as already presented in Section 5.

Table 4: Data collection guidelines

Attribute	Information captured
Case No	Listed for keeping track of numbering and for reference in other reports e.g Project Z or source control number .
Name of the reuse artefact:	The name used for artefact in the organisation e.g Test Scripts, Component X etc.
Category	Artefacts can be categorised based on the requirements of organisation. Each organisation has its own products and these products vary in different domains. This category is customisable according to the requirements of organisation e.g Code, Environments etc.
Origin	Original product from which the artefact was taken. It can be the name or version which is used inside the organisation e.g requirements specification Version 1.2.1 of Product X.
Reused in	The final product in which software artefact was reused. It can be the name or version which is used inside the organisation e.g. requirement specification Version 1.0 of Product Z.
Quarter	Depends on organisations policy for generating reports. Value of this column can be replaced by month or year. e.g 2006 Quarter 2.
Hours spent when developed in-house	Personnel hours spent on artefact when developed in-house e.g 820 hrs. In case where no historical record of personnel hours is present we may use expert opinion to estimate the number of hours.
Hours spent on reuse	Personnel hours spent on artefact when reused (in identification, understanding, integration and testing of artefact) e.g. 39 hrs.
Comment	For providing any useful comments. Used if some clarification needs to be done.

6.2 Dynamic evaluation

The next phase was to dynamically evaluate this instrument in an industrial environment, i.e. to actually use this instrument in a live setting. The dynamic validation was divided into three steps. In the first step the instrument was used at one site in one project. In the second step the instrument was used at one site in three projects and, finally, in the third step the instrument was used by several remote sites for reporting their reuse cases and the cost avoided when reusing artefacts. In total 24 projects were involved in the dynamic validation. The steps followed during the dynamic validation are described next in more detail.

6.2.1 Initiation

A tool was developed, which implemented the instrument, for managers or reuse drivers to collect data from multiple sites. This tool was in the form of a spreadsheet document with multiple worksheets, a sheet named after each individual site. An analysis, part of tool, was performed from various aspects e.g. cost avoided per site, total cost avoided at all six participating sites, artefacts which avoided costs the most, etc. The snapshot of the tool is available on web [29].The worksheets, as presented in Table 4, take care of four different categories, i.e. code reuse, document reuse (test cases, documentation, etc.), reuse of frameworks, and reuse of a development environment; however, other types can easily be added if needed. Each site has a resource (reuse driver) assigned to collect and report data back to the main site. It was noted that negligible amount of time was taken by the reuse-drivers for data collection.

6.2.2 Data Collection and Analysis

Data collection was executed on six different sites, which were geographically spread (Sweden, Netherlands, and China). An e-mail was sent to the reuse drivers of each of these sites to report the reuse cases. The data of reuse activities was collected for three consecutive months (March–May). The required data was collected and reported in time by the reuse drivers. This indicates the convenience and negligible costs associated in collection of this metric i.e. personnel-hours.

The analysis of data was carried out using the developed tool, and involved measuring the percentage savings, type of reuse artefact which was extensively reused, time taken to make the artefact reusable and total cost avoided at each individual site, as well as the combined cost avoided. The costs for data collection were not included as the projects were already completed and the amount of time spent on reusing any artefact was already known. The time taken for collecting the data was of short duration (1 personnel-hour per project).

The main aim of the dynamic validation was to judge the performance of the instrument and the focus was not on the reuse processes at the various sites. It also demonstrates the potential of what can be learned from the analysis.

6.2.3 Results from the Dynamic Validation

The results of the dynamic validation showed that the reuse of frameworks was responsible for most of the costs avoided (nearly 75% of the total savings). However, most number of reuse instances was reported from the category 'reuse of environment'. Table 5 contains the results from the dynamic validation.

The reuse of an environment took the least amount of time and development environments were the most reused artefacts. No case was reported for the reuse of documentation. The possible reasons for this could be the limited amount of time (three months) or practitioners only focusing on executable artefacts, or artefacts that are directly related to software development. The reuse of code took the most amount of time compared to the reuse of other artefacts.

Different sources, [49, 16], have claimed that it takes around 20% effort to reuse an artefact when using a black box approach. In our case, when using white box artefacts, it took, on average, around 33% effort in order to reuse it (the effort we are discussing here is in relation to the actual effort taken to develop an artefact in-house instead of reusing it). However, there were artefacts which took as much effort to reuse as it was required to develop them in-house. It was interesting to find that, in our study, the size of the artefact had no linear relation to the savings. We noted, however, that medium-sized artefacts (150–300 estimated personnel hours) required more effort to reuse, as compared to large or small artefacts. This pattern was even more evident in cases where the development environment was reused.

During the validation in industry, suggestions were made regarding the expansion of the instrument. Some stakeholders wanted it to measure cost avoided in maintenance. According to previous studies costs avoided in maintenance are the major costs avoided through software reuse [50, 51]. However, the maintenance costs can only be calculated if the reused software is used for a considerable amount of time, or can be estimated if enough historical data is recorded and thus it was decided that this aspect would be incorporated into the instrument at a later date. The solution would be to include a state (new development or maintenance) in the data collection guidelines (see Table 4).

7. DISCUSSION

This section describes different aspects of the proposed instrument, provides arguments in support of its use and also discusses potential problems, which may be present in the proposed solution.

Artefact Independent Solution: This instrument is based on all types of artefacts; basically anything created, used or stored by an organisation during software development or software maintenance. A software project can consist of several different artefacts of various types and at various stages. As most of the artefacts can be represented indirectly in terms of personnel hours, the proposed instrument can be applied on project level as well as on product level. These artefacts can vary from code-based artefact to product documentation. This was highlighted as important from our initial investigation during the observation and the survey. In particular, when using a comparable unit of measure we may compare where reuse is most beneficial with regard to the artefacts. We may also do correlation analysis to understand how the reuse of one artefact (e.g. requirements) may affect the reuse of other artefacts (e.g. frameworks and components).

Data Collection Guidelines The proposed instrument contains a data collection form (See Table 4), that can help organisations to contextualize the findings. This form contains all the necessary fields that are required to use the proposed instrument. During the dynamic evaluation, this form was used and practitioners found it effective and simple

Table 5: Results obtained from dynamic validation.

Project	Type of reused artefact	Man-hours spent to build the artefact from scratch	Reuse man-hours spent on the artefact	Total man-hours saved due to reuse	The effort required to reuse the artefact
Project 1	Code	24	8	16	33.33
Project 2	Code	24	0	24	0
Project 3	Code	40	16	24	40
Project 4	Code	60	12	48	20
Project 5	Code	80	16	64	20
Project 6	Code	160	40	120	25
Project 7	Code	250	250	0	100
Project 8	Code	1000	250	750	25
Project 9	Framework	40	2	38	5
Project 10	Framework	400	80	320	20
Project 11	Framework	500	120	380	24
Project 12	Framework	1200	200	1000	16.67
Project 13	Framework	4000	100	3900	2.5
Project 14	Environment	20	1	19	5
Project 15	Environment	30	2	28	6.67
Project 16	Environment	40	1	39	2.5
Project 17	Environment	60	10	50	16.67
Project 18	Environment	60	20	40	33.33
Project 19	Environment	60	12	48	20
Project 20	Environment	160	12	148	7.5
Project 21	Environment	160	16	144	10
Project 22	Environment	300	60	240	20
Project 23	Environment	40	20	20	50
Project 24	Framework	120	20	100	16.67

to use.

Inexpensive Data Collection The practitioners appreciated the low-cost and easy to use data collection method during evaluation. There are no complex metrics that are required to use this model. Personnel-hours spent on artefact can be found in historical data. Alternatively expert opinion can be used to estimate the hours spent on modification of artefact.

Decision Support Tool The decisions like "what to reuse ?" can be tricky for the practitioners. This instrument was not designed to do a cost-benefit analysis of reusing an artefact, but the flexibility and simplicity offered by the model enables the practitioners to use this instrument for decisions on potential reuse cases. They can use this instrument to decide on what reusable artefacts can provide most benefits with regard to cost avoidance. Similarly, organisations use this data collection form and instrument to evaluate what artefact type benefited the most in past. From Table 5 it is visible that decisions can be supported in multiple ways.

- Identify best practice: The projects with the highest amount of savings can be easily identified, and can be identified to leverage on their achieved cost avoidance.
- Identify artefacts with highest reuse potential: Identify artefacts that are more likely to give a high effort saved on reuse, such as frameworks stand out with regard to reuse savings.
- Get deeper qualitative understanding of reuse: The savings of reuse between artefacts vary greatly. Investigating the reasons for the variance in depth, and being able to explain the reasons, may provide valuable scientific results on reuse, and allows to built a knowledge base within companies as well.

Dependency on Developer's Productivity Developers' productivity can influence the personnel hours spent on various tasks. An experienced and competent developer can integrate a component in less time compared to a new and inexperienced developer. The same assumption can be applied if the artefact is developed in-house; an inexperienced developer might require more time to develop the artefact, compared to the experienced developer. But, according to [12], the experience level of a developer does not affect software reuse costs to a high extent.

Cost Avoidance in Software Product Lines Software product lines consist of set of product sharing similar features developed for the need of specific market [14]. Software product lines require prior investment in the form of development of reusable resources [52]. Software reuse cost avoidance implies that no resources are expended before the reuse instance. The return on investment (ROI) reuse cost models are better suited to calculate the benefits of reuse in software product lines [52]. However, this instrument can be used to calculate the cost avoidance of a reuse instance in software product lines.

Organisational Practices: The following are the assumptions regarding organisational practices that would allow a given organisation to apply the instrument: The organisation needs to decide on a quantitative way to measure personnel hours in a consistent way. The measure of personnel hours is related to the nature of the task, i.e., a difficult or a newer task is expected to take more personnel hours. Secondly, the organisation needs to decide the level of detail to measure personnel hours, e.g., is it the actual time spent working on an artefact is important or other activities like searching, identifying, integrating and testing the artefact?

An organisation can easily extend the instrument to use all these different activities e.g. by using separate variables for identifying, searching, modifications of artefacts. Thirdly, if multiple personnel are using the same artefact, having different rates for personnel hours (H), the total cost avoided is simply the sum of the cost avoided by individual persons.

Comparison with related work: As far as we can tell, only a few previous studies exist, and they are based on lines of code in two out of three cases [16, 33]. Hence, we claim our proposed instrument to be better in terms of coverage of various reusable artefacts. A third instrument, [34], related to CA, is quite similar to our proposed instrument. However, in terms of the number of variables to be considered the instrument proposed in this paper appeared to be easier to apply in terms of the metrics it requires. Furthermore, guidelines for collection of metrics for measuring reuse cost avoidance were proposed by our study.

Tool implementing the Instrument: The tool supporting the instrument helped in the analysis of reuse cases. The practitioners were able to compare the results of different sites and identify the sites that had better or worse reuse cases. Different reused artefacts were analysed and compared with each to identify the artefacts that avoided the maximum costs. Furthermore, the tool generated the graphs and chart-based reports for the stakeholders to reflect on the costs avoided through reuse.

8. VALIDITY THREATS

SLR: In our SLR there were studies of which no full text was available. These studies were sometimes old and, thus, hard to locate. This can affect the validity of the findings from the SLR and [37] has already reported this problem in literature and its effects. Our study focused on models that only deal with cost avoidance through reuse. There are models that can calculate return on investment, cost-benefit analysis and CA within a single solution (in-directly), even though they do not mention the term cost avoidance. However, these models are complex to apply and the metrics are hard to collect. For these reasons we did not include them in this study. The bias in the SLR inclusion criteria and exclusion criteria may lead to a case where a study is excluded even though it should have been part of final results.

It is also apparent that, even though the corpus of studies was large in the beginning, only very few studies reported cost avoidance models, which also points to a research gap and the need to address the restrictions of existing models, such as their focus on lines of code.

We believe that the SLR provides a good sample of cost avoidance models in software engineering since we did an exhaustive search and, in addition, scanned reference lists on all publications for any additional papers to include in the study.

Industrial Evaluation: The threats to the external validity were reduced by involving the industry practitioners. Our proposed instrument was dynamically validated in one organisation only; however, since personnel hours can be considered to be a basic metric, collected by most organisations, the instrument would be easy to apply in other organisations. The instrument was, nevertheless, tested on 24 projects at 6 geographically distributed sites. This shows a reasonable level of validation considering two factors: (i) The details of industrial evaluation for two CA models were not sufficient while the third model did not describe industrial validation. (ii) Industrial validation is generally a hard and a time-consuming task.

9. CONCLUSIONS

This study contributes to the knowledge of software reuse cost avoidance. Varieties of reuse economic models are found in literature, but only a few studies address cost avoidance aspects of software reuse. In this study we try to address the difficulties related to the measurement of cost avoidance by providing an instrument that is simpler to use and, in addition, covers several types of reuse artefacts other than code reuse.

A systematic literature review was conducted to identify and analyse cost models that can measure cost avoidance through reuse. Three models were classified as reuse cost avoidance models. Two of the models were based on lines of code. The third model was *artefact-independent* but it was not evaluated in industry yet. This model required complex metric such as 'search hit-rate', ' time to understand the suitability of each potentially reusable item' etc. The conclusion of systematic literature review was that there is lack of literature on measurement of reuse cost avoidance and there is only one *artefact-independent* model for measuring cost avoidance. None of the three models described the guidelines for collecting metrics used in each model.

An instrument to measure reuse cost avoidance and guidelines on what to measure for cost avoidance were proposed. This instrument can measure costs avoided from all kinds of commonly reused artefacts. Two metrics are required in order to apply this instrument. The first metric is the time spent on an artefact when it is developed in-house and the other metric is the time spent on an artefact in order to reuse it. Static and dynamic validation of the instrument was conducted in industry. Six different sites, in three countries, were contacted to report their reuse cases using the implemented tool. There were in total 24 projects participating. The instrument performed according to the expectations and users were satisfied with the results. The instrument, with its accompanying tool and documentation has been transferred to industry.

The instrument was *artefact-independent* and uncomplicated essence of it makes it useful for the software industry. Secondly, instrument can be used as a decision support tool for deciding on what to reuse or which reuse artefact can give maximum benefits. The dynamic validation pointed out that the instrument should preferably also accommodate the costs avoided in maintenance, when reusing artefacts. However, this requires a longitudinal study to collect further data. The scope of this study was limited to the direct savings through reused artefacts; but in the future this instrument could include savings in maintenance when reusing artefacts.

10. ACKNOWLEDGEMENTS

The research presented in this paper was partly funded by the Swedish Knowledge Foundation, and the project "Professional Licentiate of Engineering Research School", conducted at Blekinge Institute of Technology, Software Engineering Research Lab - Sweden.

11. REFERENCES

[1] Tiwari, R. and Goel, N., 2013. Reuse: reducing test effort. ACM SIGSOFT Software Engineering Notes, 38(2), pp.1-11.

[2] Mohagheghi, P. and Conradi, R., 2008. An empirical investigation of software reuse benefits in a large telecom product. ACM Transactions on Software Engineering and Methodology (TOSEM), 17(3), p.13.

[3] Bassett, P.G., 1998, June. How to solve the reuse problem. In icsr (p. 373). IEEE.

[4] Mohagheghi, P. and Conradi, R., 2007. Quality, productivity and economic benefits of software reuse: a review of industrial studies. Empirical Software Engineering, 12(5), pp.471-516.

[5] Sherif, K., Zmud, R.W. and Browne, G.J., 2006. Managing peer-to-peer conflicts in disruptive information technology innovations: The case of software reuse. MIS quarterly, pp.339-356.

[6] Frakes, W.B. and Succi, G., 2001. An industrial study of reuse, quality, and productivity. Journal of Systems and Software, 57(2), pp.99-106.

[7] Nazareth, D.L. and Rothenberger, M.A., 2004. Assessing the cost-effectiveness of software reuse: a model for planned reuse. Journal of Systems and Software, 73(2), pp.245-255.

[8] Frakes, W. and Terry, C., 1996. Software reuse: metrics and models. ACM Computing Surveys (CSUR), 28(2), pp.415-435.

[9] Krueger, C.W., 1992. Software reuse. ACM Computing Surveys (CSUR), 24(2), pp.131-183.

[10] Holmes, R., 2008. Pragmatic software reuse (Doctoral dissertation, University of Calgary).

[11] Holmes, R. and Walker, R.J., 2007, May. Supporting the investigation and planning of pragmatic reuse tasks. In Proceedings of the 29th international conference on Software Engineering (pp. 447-457). IEEE Computer Society.

[12] Frakes, W.B. and Fox, C.J., 1995. Sixteen questions about software reuse. Communications of the ACM, 38(6), pp.75-ff.

[13] Selby, R.W., 2005. Enabling reuse-based software development of large-scale systems. Software Engineering, IEEE Transactions on, 31(6), pp.495-510.

[14] Pohl, K., Böckle, G. and van Der Linden, F.J., 2005. Software product line engineering: foundations, principles and techniques. Springer Science & Business Media.

[15] Lim, W.C., 1996, April. Reuse economics: A comparison of seventeen models and directions for future research. In Software Reuse, 1996., Proceedings Fourth International Conference on (pp. 41-50). IEEE.

[16] J. Poulin, 1996, Measuring software reuse: Principles, practices, and economic models. Boston, MA, USA Addison-Wesley Longman Publishing Co., Inc.

[17] Konda, B.M. and Mandava, K.K., 2010. A systematic mapping study on software reuse.

[18] Ashenbaum, B., 2006. Defining cost reduction and cost avoidance. CAPS Research Critical Issues Report.

[19] Woods, J.E., 1988. Cost avoidance and productivity in owning and operating buildings. Occupational medicine (Philadelphia, Pa.), 4(4), pp.753-770.

[20] Mutnick, A.H., Sterba, K.J., Peroutka, J.A., Sloan, N.E., Beltz, E.A. and Sorenson, M.K., 1997. Cost savings and avoidance from clinical interventions. American Journal of Health-System Pharmacy, 54(4), pp.392-396.

[21] Miyagawa, C.I. and Rivera, J.O., 1986. Effect of pharmacist interventions on drug therapy costs in a surgical intensive-care unit. American Journal of Health-System Pharmacy, 43(12), pp.3008-3013.

[22] Laszlo, G.P., 1997. The role of quality cost in TQM. The TQM Magazine, 9(6), pp.410-413.

[23] Gray, D. and Steenhuis, H.J., 2003. Quantifying the benefits of participating in an industry university research center: An examination of research cost avoidance. Scientometrics, 58(2), pp.281-300.

[24] Jacobson, I., Griss, M. and Jonsson, P., 1997. Making the reuse business work. Computer, 30(10), pp.36-42.

[25] Guerrieri, E., Lashway, L.A. and Ruegsegger, T.B., 1989, July. An acquisition strategy for populating a software reuse library. In National Conference on Software Reusability (pp. 19-20).

[26] Lim, W.C., 1994. Effects of reuse on quality, productivity, and economics. Software, IEEE, 11(5), pp.23-30.

[27] Kitchenham, B. and Charters, S., Guidelines for performing systematic literature reviews in software engineering. 2007. URL http://www. dur. ac. uk/ebse/resources/Systematic-reviews-5-8. pdf.

[28] Dybå, T. and Dingsøyr, T., 2008. Empirical studies of agile software development: A systematic review. Information and software technology, 50(9), pp.833-859.

[29] "Cost Avoidance SLR - Supplementary Information". , 2016. Web. 13 Apr. 2016. https://sites.google.com/site/mohsinirshad390/pub/CostAvoidanceSLR.pdf

[30] Goodman, L.A., 1961. Snowball sampling. The annals of mathematical statistics, pp.148-170.

[31] Poulin, J.S., Caruso, J.M. and Hancock, D.R., 1993. The business case for software reuse. IBM Systems Journal, 32(4), pp.567-594.

[32] Poulin, J.S. and Caruso, J.M., 1993, March. A reuse metrics and return on investment model. In Software Reusability, 1993. Proceedings Advances in Software Reuse., Selected Papers from the Second International Workshop on (pp. 152-166). IEEE.

[33] *Software reuse metric plan*, Defense Information Systems Agency, Falls Church, VA, August 1993, dISA/JIEO/CIM.

[34] Amar, L. and Coffey, J., 2005. Measuring the benefits of software reuse-examining three different approaches to software reuse. Dr Dobbs Journal, 30(6), pp.73-76.

[35] Offen, R.J. and Jeffery, R., 1997. Establishing software measurement programs. Software, IEEE, 14(2), pp.45-53.

[36] McCarey, F., ò Cinneide, M. and Kushmerick, N., 2008. Knowledge reuse for software reuse. Web Intelligence and Agent Systems: An International Journal, 6(1), pp.59-81.

[37] Jacobs, D., 2005. Accelerating process improvement using agile techniques. CRC Press.

[38] Demirors, O. and Gencel, C., 2009. Conceptual association of functional size measurement methods. IEEE software, 26(3), p.71.

[39] Morozoff, E.P., 2010. Using a line of code metric to understand software rework. Software, IEEE, 27(1),

pp.72-77.

[40] Gorschek, T., Wohlin, C., Carre, P. and Larsson, S., 2006. A model for technology transfer in practice. Software, IEEE, 23(6), pp.88-95.

[41] Petersen, K. and Wohlin, C., 2009, October. Context in industrial software engineering research. In Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement (pp. 401-404). IEEE Computer Society.

[42] Dyba, T., 2013. Contextualizing empirical evidence. Software, IEEE, 30(1), pp.81-83.

[43] Kanso, A. and Monette, D., 2014, September. Foundations for long-term collaborative research. In Proceedings of the 2014 international workshop on Long-term industrial collaboration on software engineering (pp. 43-48). ACM.

[44] Gencel, C., Petersen, K., Mughal, A.A. and Iqbal, M.I., 2013. A decision support framework for metrics selection in goal-based measurement programs: GQM-DSFMS. Journal of Systems and Software, 86(12), pp.3091-3108.

[45] Poulin, J. S. (1993). Issues in the development and application of reuse metrics in a corporate environment. In Fifth International Conference on Software Engineering and Knowledge Engineering..

[46] Jørgensen, M., 2004. A review of studies on expert estimation of software development effort. Journal of Systems and Software, 70(1), pp.37-60.

[47] Hughes, R.T., 1996. Expert judgement as an estimating method. Information and Software Technology, 38(2), pp.67-75.

[48] Rush, C. and Roy, R., 2001. Expert judgement in cost estimating: Modelling the reasoning process. Concurrent Engineering, 9(4), pp.271-284.

[49] Mili, A., Chmiel, S.F., Gottumukkala, R. and Zhang, L., 2001. Managing software reuse economics: an integrated ROI-based model. Annals of Software Engineering, 11(1), pp.175-218.

[50] Balda, D. and Gustafson, D.A., 1990. Cost estimation models for reuse and prototype SW development life-cycles. ACM SIGSOFT Software Engineering Notes, 15(3), pp.42-50.

[51] Tomer, A., Goldin, L., Kuflik, T., Kimchi, E. and Schach, S.R., 2004. Evaluating software reuse alternatives: a model and its application to an industrial case study. Software Engineering, IEEE Transactions on, 30(9), pp.601-612.

[52] Böckle, G., Clements, P., McGregor, J.D., Muthig, D. and Schmid, K., 2004. Calculating ROI for software product lines. Software, IEEE, 21(3), pp.23-31.