A Study of Developer Attitude to Component Reuse in Three IT Companies

Jingyue Li[1], Reidar Conradi[1,3], Parastoo Mohagheghi[1,2,3], Odd Are Sæhle[1], Øivind Wang[1], Erlend Naalsund[1], and Ole Anders Walseth[1]

[1] Department of Computer and Information Science,
Norwegian University of Science and Technology (NTNU),
NO-7491 Trondheim, Norway
{jingyue, conradi}@idi.ntnu.no
[2] Ericsson Norway-Grimstad, Postuttak, NO-4898 Grimstad, Norway
{parastoo.mohagheghi}@ericsson.com
[3] Simula Research Laboratory, P.O.BOX 134, NO-1325 Lysker, Norway

Abstract. The paper describes an empirical study to investigate the state of practice and challenges concerning some key factors in reusing of in-house built components. It also studies the relationship between the companies' reuse level and these factors. We have collected research questions and hypotheses from a literature review and designed a questionnaire. 26 developers from three Norwegian companies filled in the questionnaire based on their experience and attitudes to component reuse and component-based development. Most component-based software engineering articles deal with COTS components, while components in our study are in-house built. The results show that challenges are the same in component related requirements (re)negotiation, component documentation and quality attributes specification. The results also show that informal communications between developers are very helpful to supplement the limitation of component documentation, and therefore should be given more attention. The results confirm that component repositories are not a key factor to successful component reuse.

1. Introduction

Systematic reuse is generally recognized as a key technology for improving software productivity and quality [17]. With the maturity of component technologies, more and more companies have reused their software (program) in the form of components. Component reuse consists of two separate but related processes. The first deals with analysis of the application domain and development of domain-related components, i.e. *development-for-reuse*. The second process is concerned with assembling software system from prefabricated components, i.e. *development-with-reuse*. These two processes are tightly related, especially in reusing in-house built components. The number of components and the ratio of reused components to total components will determine the reuse benefits (e.g. improved productivity and quality) [11][23].

To investigate the current state of practice and challenges for *development-with-reuse* in the IT industry, and to investigate the relationship between companies' reuse level and some key factors in reusing of in-house components, an empirical study was performed as part of two Norwegian R&D projects. These projects were SPIKE (Software Process Improvement based on Knowledge and Experience) [29] and INCO (INcremental and COmponent-based development) [30]. From the literature review, we defined several research questions and hypotheses. A questionnaire was designed to investigate these questions. Developers from three Norwegian IT companies filled in the questionnaire based on their experience and attitudes to component reuse.

From the results of the survey, we found some new challenges in component reuse and component-based development based on in-house built components. The results support some commonly held beliefs and contradict others.

As the sample size of current research is still small, this study cannot provide statistically significant tests on hypotheses, and is therefore a pre-study. Later studies will be undertaken with refined hypotheses and on a larger sample.

The reminder of the paper is structured as follows: Section 2 presents some general concepts. Section 3 describes the research approach. Section 4 presents the survey results. Section 5 gives a detail discussion on the survey results. Conclusion and future research are presented in section 6.

2 Component reuse and component-based development

Software reuse can take many different forms, from ad-hoc to systematic [16]. In the broad definition of reuse, it includes reusing everything associated with software projects, such as procedures, knowledge, documentation, architecture, design and code. In our research, we focus on systematic reuse of software code. The code reuse literature has identified reuse practice and success factors through several case studies and surveys. A major reuse effort is the REBOOT (Reuse Based on Object-Oriented Techniques) consortium [25]. This effort was one of the early reuse programs that recognized the importance of not only the technical, but also the organizational aspects of reuse [18]. As more experience become available from industrial studies, non-technical factors, such as organization, processes, business drivers and human involvement, appeared to be at least as important as technological issues [15][19].

Following the success of the structured design and OO paradigms, component -based software development has emerged as the next revolution in software development [27]. More and more IT companies have started to reuse code by encapsulating it into components. Whitehead defines a component as: *A software component is a separable piece of executable software, which makes sense as a unit, and can interoperate with other components, within some supporting environment. The component is accessible only via its interface and is capable of use 'as-is', after any necessary installation and configuration procedures have been carried out* [28].

Component-based development is assumed to have many advantages. These include more effective management of complexity, reduced time to market, increased productivity, improved quality, a greater degree of consistency and a wider range of usability [4][13]. It also brings many challenges, because it involves various stakeholders and roles, such as component developers, application developers, and customers. Different stakeholders and roles have different concerns [3], and face different issues and risks [2][27].

Component-based development differs from traditional development, where the usual approach is for stakeholders to agree upon a set of requirements and then build a system that satisfies these requirements from scratch. Component-based development builds application by reusing existing components. Available components may not be able to satisfy all the requirements. Therefore, component-based projects must have flexibility in requirements, and must be ready to (re)negotiate the requirements with the customer. Moreover, components are intended to be used 'as-is'. If some additional functionality is required, 'glue-code' is needed to be built to meet the differences between the requirement and component functionality. Another important feature of component-based development is the strong focus on the quality attributes (such as reliability, performance, and security etc.) and related testing. A major effort has to be put into checking how components perform, how well they interact, and to make sure that they are indeed compatible. Components may be developed in-house, acquired as COTS (commercial-off-the-shelf) [3], or even as OSS (Open Source Software) [5]. Most current research on component-based software engineering focuses on COTS-based development. Because COTS users cannot access the source code and must rely on vendors to give technical support, COTS-based development is assumed to be more challenging. Therefore, there is little research on the challenges based on in-house built components.

3 Research approach

The difference between development based on in-house built components and development based on COTS is that the former is related very tightly with *development-for-reuse*. Component reuse is generally an incremental procedure. The company will build some reusable components in the beginning. In case of successful reuse, more and more code will be encapsulated into reusable components. The more reusable components are developed, the more complex will the development process be, and more support is required from the organization [8]. Our motivation is to investigate the relationship between companies' reuse level and some key factors in component-based development so that company with low reuse level can make necessary software process improvements when moving to a higher reuse level.

3.1 Research questions

To reuse in-house components successfully, developers must follow three basic steps [19]:

- Formulate the requirements in a way that supports retrieval of potentially useful reusable components.
- Understand the retrieved components.

– If the retrieved components are sufficiently 'close' to the needs at hand and are of sufficient quality, then adapt them.

From these steps, we selected several key factors. For step 1, we focus on the efficiency of component related requirements (re)negotiation and the value of component repository. For step 2, we study how knowledge about components can be transferred from a component provider to a component user. For step 3, our study focuses on definition and reasoning of quality attributes of components.

There is little research on the need for requirements (re)negotiation when components are built in-house. People assume that owning source code of in-house built components allows them to do any changes to meet the customers' requirements. However, components are intended to be used 'as-is', even it is built in-house. So, our first research question is:

RQ1. Does requirements (re)negotiation for in-house components really work as efficiently as people assume?

Crnkovic et al. have proposed that to successfully perform the component-based requirements (re)negotiation, a vast number of possible component candidates must be available, as well as tools for finding them [9]. Companies with a higher reuse level usually have more component candidates, more experience, and better experience than companies with a lower reuse level. So, our second research question is:

RQ2. Does the efficiency of component related requirements (re)negotiation increase with more in-house built components available?

To investigate this question, we formalized a null hypothesis H01 and an alternative hypothesis HA1 as follows:

H01. There is no relationship between the companies' reuse level and the efficiency of component related requirements (re)negotiation.

HA1. There is a positive relationship between the companies' reuse level and the efficiency of component related requirements (re)negotiation.

Concerning a component repository, Frakes claimed that it should not be given much attention, at least initially [12]. So, our third research question is:

RQ3. Does the value of component repository increase with more reusable components available?

To investigate this opinion more deeply, a null hypothesis H02 and an alternative hypothesis HA2 was proposed:

H02. There is no relationship between the companies' reuse level and the value of component repository.

HA2. There is a positive relationship between the companies' reuse level and the value of component repository.

A complete specification of a component should include its functional interface, quality characteristics, use cases, tests, etc. While current component-based technologies successfully manage functional interfaces, there is no satisfactory support for expressing quality parts of a component [9]. So, our fourth research question is:

RQ4. How can a component user acquire sufficient information about relevant components?

Berglund claimed that growing reusable software components will create a new problem, i.e. the *information-overload problem*. Therefore, learning which component to use and how to use them become the central part of software development [1]. Companies with a higher reuse level usually have more reusable components than companies with lower reuse level. So, our fifth research question is:

RQ5. Does the difficulty of component documentation and component knowledge management increase with increasing reuse level?

To study this question, we formalize null hypothesis H03 and alternative hypothesis HA3:

H03. There is no relationship between the companies' reuse level and developers' satisfaction with component documentation.

HA3. There is a negative relationship between the companies' reuse level and developer' satisfaction with component documentation.

One key issue in component-based development is *trust*, i.e. we want to build trustworthy systems out of parts for which we have only partial knowledge [7]. Current component technologies allow systems builders to plug components together, but contribute little to ensure how well they will play together or to fulfill certain quality properties. So, the sixth research question is:

RQ6. Do developers trust the quality specification of their in-house built components? If the answer is no, how can they solve this problem?

3.2 The questionnaire

The questionnaire included five parts. The questions in the first part were used to investigate the *reuse level* of the companies. The definition of *reuse level* in this study is the number of reused components vs. the number of total components in the organization. The other four parts were organized based on the four key factors. Each question in the questionnaire was used to study one or more research questions. The details of questions are showed in the following Table 1. The correspondences between the questions in the questionnaire and research questions are showed in Table 2. To increase the reliability of our survey, the questionnaire also included the definition of concepts used in the questionnaire, and the questions about the respondents' personal information.

3.3 Data collection

The study was performed in three Norwegian IT companies. Data collection was carried out by NTNU PhD and MSc students. Mohagheghi, Naalsund, and Walseth performed the first survey in Ericsson in 2002. In 2003, Li, Sæhle and Wang performed the survey reusing the core parts of the questionnaire in two other companies (i.e. EDB Business Consulting and Mogul Technology). We selected those three companies because they have experience on component reuse and would like to cooperate with NTNU in this research. The respondents are developers in these three companies. They answered the questionnaires separately. The questionnaires were filled in either by hand or electronically (as a Word file). The MSc students provided support with possible problems in answering the questionnaire.

Table 1. Questions in the questionnaire

Reuse level
Q1. What is the reuse level in your organization? Q2. To what extend do you feel affected by reuse in your work?
Component related requirements (re)negotiation
Q3. Are requirements often changed/ (re)negotiated in typical develop projects? Q4. Are requirements usually flexible in typical projects? Q5. Do the component related requirements (re)negotiation processes work efficiently in typical projects?
Value of component repository
Q6. Would the construction of a reuse repository be worthwhile?
Component understanding
Q7. Do you know the architecture of the components well? Q8. Do you know the interface of the components well? Q9. Do you know the design rules of the components well? Q10a. Is the existing design/code of reusable components sufficiently documented? Q10b. If the answer of Q10a is ‘sometimes’ or ‘no’, is this a problem? Q10c. If the answer of Q10a is ‘sometimes’ or ‘no’, what are the problems with the documentation? Q10d. If the answer of Q10a is ‘sometimes’ or ‘no’, how would you prefer the documentation? Q10e. What is your main source of information about reusable components during implementation? Q10f. How do you decide whether to reuse a component ‘as-is’, ‘reuse with modification’ or ‘make a new one from scratch’?
Quality attributes specification of components
Q11. Are specifications for components’ quality attributes well defined? Q12. Do you test components after modification for their quality attributes before integrating them with other components?

Table 2. Correspondence between Questions in the questionnaire and Research Questions

	RQ1	RQ2	RQ3	RQ4	RQ5	RQ6
Q1-Q2		X	X		X	
Q3-Q5	X	X				
Q6			X			
Q7-Q10f				X	X	
Q11-Q12						X

Below, we briefly characterize these three companies and respondents.

3.3.1 Companies

Ericsson Norway-Grimstad started a development project five years ago and has successfully developed two large-scale telecommunication systems based on the same architecture and many reusable components in cooperation with other Ericsson organization. Their two main applications share more than 60% of ca. 1M lines of code [22].

EDB Business Consulting in Trondheim (now Fundator) is an IT-consultant firm which helps its customers to utilize new technology. It started to build reusable components from 2001. They have built some reusable components based on the Microsoft .Net in their eCportal framework (i.e. a web-application framework) 1.0 & 2.0. These components have been successfully reused in their new e-commence applications.

Mogul Technology (now Kantega) in Trondheim has large customers in the Norwegian finance- and bank sector. The main responsibilities are development and maintenance of the customers' Internet bank application. The application was originally a monolithic system. After several years in production, the customer itself took initiative to reengineer the old system to a component-based solution based on EJB component model in 2002. At the time of the survey, some components have been created and reused in their new Internet bank system.

3.3.2 Respondents

There were 200 developers at Ericsson in Grimstad, where we sent out 10 questionnaires to developers in one development team and got 9 filled-in questionnaires back. There were 20 developers in EDB Business Consulting in Trondheim, and we gathered 10 filled-in questionnaires back out of 10. We distributed 10 questionnaires to 22 developers at Mogul Technology in Trondheim and got 7 back. Those developers were selected because their work was related to component reuse, and they could assign effort to participate in the survey. This is non-probability sampling, which is based on convenience. Most participants in this survey have a solid IT background. 6 of 26 respondents have MSc degree in computer science and all others have a bachelor degree in computer science or telecommunication. More that 80% of them have more than 5 years of programming experience. The details of their position and their experience in the current organization are summarized in the following Table 3.

4 Survey Results

In this section, we summarize the result of the survey. All the following statistical analyses are based on valid answers, i.e. *Don't Know* answers are excluded. The statistical analysis tool we used is SPSS Version 11.0.

4.1 Different reuse level in these companies

First, we wanted to know the reuse level in those three companies. Q1 and Q2 were asked to get the answer based on developers' subjective opinion on this issue. The result of Q1 is showed in Fig. 1, and the result of Q2 is showed in Fig. 2. From Fig. 1 and Fig. 2, we can see that most developers in Ericsson think that the reuse level in their company is very high or high. Most developers in EDB regard the reuse level in their company is high or medium. Most developers in Mogul think that the reuse level in their company is medium or little.

Table 3. Background of the respondents

Company	Position and working experience in the organization
Ericsson Norway-Grimstad	2 system architects, 7 designers. 1 person has 13 years of experience 7 persons have experience from 2-5 years, 1 person has 9 months of experience,
EDB Business Consulting in Trondheim	1 project manager, 5 developers and 4 IT consultants. 1 person has 17 years of experience 8 persons have experience from 3-8 years, 1 person has 2 years of experience.
Mogul Technology in Trondheim	6 developers and 1 maintainer (previous developer). 1 person has 10 years of experience, 6 persons have experience from 2-5 years.

4.2 Component related requirements (re)negotiation

Questions Q3-Q5 were asked to investigate RQ1.We can see that no respondents to Q3 believe that the requirements were never changed/ (re)negotiated. Only 8% of respondents to Q4 think the requirements of their typical project are not flexible. However, only 48% of respondents to Q5 think component related requirements (re)negotiation works well. To study RQ2 and test the hypothesis H01, the correlation between the reuse level and response to Q5 is studied. We assign ordinal values to Ericsson, EDB and Mogul to represent their different reuse levels based on the responses to Q1 and Q2 (Ericsson = 3, EDB = 2, Mogul = 1). We also assign ordinal value to the answer of Q5 (Yes = 3, Sometimes = 2, No =1). The result of correlation between them using one-tailed *Spearman Rank Correlation Coefficient* analysis is .112, and the significance is .306. This shows that there is no significant statistical relationship between the reuse level and the efficiency of component related requirements (re)negotiation.

4.3 Value of component repository

From the answer of Q6, we found that 71% of respondents in Mogul and EDB regard constructing a component repository as worthwhile, against 57% in Ericsson. To study RQ3 and test hypothesis H02, the relationship between the answer of Q6 and the reuse level is studied. We use the same ordinal number mapping as previously. The result of correlation between them using one-tailed *Spearman Rank Correlation Coefficient* analysis is -.124, and significance is .297, which shows that there is no obvious relationship between them.

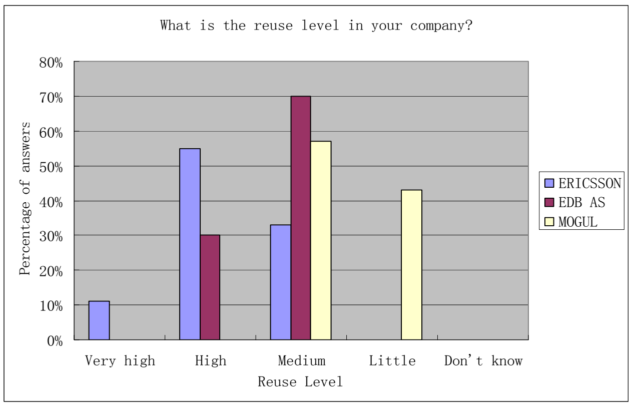

Fig. 1. Result of the question “What is the reuse level in your company?”

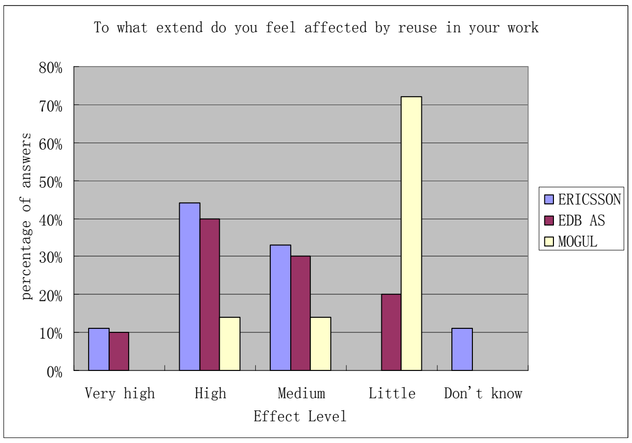

Fig. 2. Result of the question “To what extend do you feel affected by reuse in your work?”

4.4 Component understanding

Questions Q7-Q10f were used to investigate RQ4. For Q7, Q8 and Q9, the results show that 67% of the respondents think the component structure is well understood, 61% say that the component interfaces are understood, and 63% regard the design rules of components are also well understood. But for the responses to question Q10a, no one thinks that the design/code of components is well documented, 73% think that they are sometimes well defined, and 27% believe that they are not well documented.

Furthermore, the answers to questions Q10b and Q10c indicate that 86% believe that insufficient component documentation is a problem, e.g. documentation is not complete, not updated, and difficult to understand, etc. From responses to Q10d and Q10f, we can see that the preferable way of documentation is web pages. Some of the developers' knowledge of how to use components comes from informal communication sources, for example, previous experience, suggestions from local experts, etc. To study RQ5 and test hypothesis H03, the association between reuse level and response to Q10a is studied. We use the same ordinal number mapping as previously. The result of correlation between them using one-tailed *Spearman Rank Correlation Coefficient* analysis is -.469, and significance is .014, which shows that there is a weak negative relationship between them. It means that the higher the companies' reuse level, the less satisfied a developer is with the component documentation.

4.5 Quality attributes of components

Question Q11 and Q12 were used to investigate RQ6. From the responses to these questions, we see that 70% of the participants regard the design criteria for quality requirements are not well defined, and 87% will test the quality attributes of components after component modification, before integrating them into the system.

5 Discussions

Based on the result of the survey, we discuss our research questions and hypotheses, and discuss the limitations and threats to validity.

5.1 Component related requirements (re)negotiation

Much research focus on how to improve the efficiency of component related requirements (re)negotiation in COTS-based development [20][24][26]. The main reason is that people think the challenges in requirements (re)negotiation are due to the lack of access to source code, to timely vendor supports, or to the lack of engineering expertise to modify the integrated components [26]. In our case, the components are mostly built in-house. The above constrains on COTS components are not considered as challenges with built in-house components. From the responses to question Q3-Q5, we found that although 92% think that requirements of their typical projects are flexible, less than half think the component related requirements (re)negotiation in their typical projects works well.

Since components are intended to be used 'as-is', it is possible that an in-house reusable component meeting all the requirements will not be found. So, even though the components are built in-house, requirements (re)negotiation is necessary. For research question RQ1, we do not want to claim that the requirements (re)negotiation based on in-house components is more difficult than COTS-based components. We just want to emphasize that requirements (re)negotiation based on in-house components is also important but not efficient.

From the test result on H01, we cannot find a statistically significant relationship between the reuse level and the efficiency of component related requirements (re)negotiation. So, we cannot reject null hypothesis H01. Our conclusion to RQ2 is that when IT companies change from a low reuse level to a higher reuse level, they probably cannot expect that component-based requirements (re)negotiation becomes easier and more efficient.

5.2 Component repository

Some researchers have claimed that repository is important, but not sufficient for successful reuse [18][21]. Our data confirms that developers are positive, but not strongly positive to the value of component repository. So, this result gives future support to the previous conclusion.

From the test result on H02, we can see that there is no statistically significant relationship between developers' positive attitude to a component repository and reuse level. So, we cannot reject null hypothesis H02. Our conclusion to RQ3 is that companies are not expected to invest in a repository to increase reuse.

5.3 Component understanding

Transferring component knowledge from the component developer to the component user is critical for successful component reuse. The answers of Q7-Q9 show that most developers understand the components in detail. However, the answers of Q10a-Q10c show that no one believes that the components are well documented because the documents are either incomplete or not updated. So, our question is *"How can developers still understand the components without good documentation?"* From the answers to question Q10e and Q10f, we found that most developers got the knowledge of components from informal channels, such as previous experience and local experts. The most important feature of a component is the separation of its interface from its implementation. The component implementation is only visible through its interface. Moreover, current component documentation technologies cannot describe all the information the developer required, such as performance, reliability, and security etc. Therefore, informal knowledge transfer should be considered to supplement the insufficiency of formal component documentation and specification. This point was showed in other empirical studies as well [6][10]. For research question RQ4, we found that informal knowledge transfer is especially important in the component reuse. One possible solution is to have special interest groups or mailing lists for a components (or group of similar components) so that component users can share knowledge and experience of component usage.

From the test result on H03, we found a weak negative relationship between reuse level and developers' satisfaction with the documentation. We reject the null hypothesis H03 and accept the alternative hypothesis HA3, It means the higher the companies' reuse level, the less satisfied a developer is with components' documentation. Marcus et al. concluded that combine reuse education and training provided for staff with other reuse activity can lead to all the success of reuse [18]. Our conclusion to RQ5 implies that when a company moves from a low reuse level to high level, more effort should be spent on the component documentation and component knowledge management.

5.4 Quality attributes of components

Component-based development relies on the availability of high quality components to fill roles in a new intended system. When components are created or changed, we must ensure that they do not only fulfill the functional requirements, but also quality requirements. For research question RQ6, we found that most developers are not satisfied with the specification of components' quality attributes and therefore cannot use this information. Therefore, how can we model quality properties of both components and systems, and reason about them, particularly in the early stage of system development is still a key challenge in component-based development.

5.5 Threats to validity

We now discuss the possible validity threats in this study. We use the definition given by Judd et al. [14].

Construct validity In our case, the main construct issue applies to the variables chosen to characterize the data set. The independent variable, i.e. reuse level, is the most sensible one. The results of questions Q1 and Q2 give a qualitative and consistent value on this variable.

Internal validity A major threat to this validity is that we have not assessed the reliability of our measurement. Most variables are measured on a subjective ordinal scale. An important issue for future studies is to ensure the reliability and validity of all measurement. In this survey, we gave clearly specified concepts in the questionnaire and provided support to possible misunderstanding. These methods partly increased the reliability.

External validity The small sample size and lack of randomness in the choice of companies and respondents are threats to external validity. In general, most empirical studies in industry suffer from non-representative participation, since companies that voluntarily engage in systematic improvement activities must be assumed to be better-than-average.

Conclusion validity This study is still a pre-study. Future studies will be implemented to give more statistically significant results.

6 Conclusion and future work

This study has investigated challenges related to four key factors for development based on in-house components, especially in development-with-reuse. These factors are component related requirements (re)negotiation, component repository, component understanding and components' quality attribute specification. Another contribution is that we compared three IT companies with different reuse levels to study the possible trend and challenges in these factors when more and more code will be encapsulated as reusable components inside a company.

– For *component-based requirements (re)negotiation*, the results of research questions RQ1 and RQ2 show that requirements (re)negotiation for in-house built components is important but not efficient. The efficiency will probably not increase with higher reuse level.
– For the *component repository*, the results of research question RQ3 confirm that a component repository is not a key factor for successful reuse. Furthermore, the potential value of a component repository will probably not increase with higher reuse levels.
– For *component understanding*, the results of research questions RQ4 and RQ5 show that most developers are not satisfied with the component documentation, and developers' satisfaction with component documentation will probably decrease with higher reuse level. The results also show that informal communication channels, which developers can get necessary information about the components through, should be given more attention
– For *components' quality attribute specification*, the result of research question RQ6 shows that developers still need to spend much effort on testing, as they cannot get relevant information from component specifications.

The main limitation of our survey is that it depends on the subjective attitudes of developers, and with few companies and participants involved. Later studies are planned to be undertaken with more precise quantitative methods and on more companies with more distinct reuse levels. Case studies will also be undertaken to follow the change of companies from lower reuse level to higher reuse level to future investigate our research questions.

7 Acknowledgements

This study is supported by the SPIKE and INCO projects. We thank the colleagues in these projects, and all the participants in the survey.

References

1. Erik Berglund: Writing for Adaptable Documentation. Proceedings of IEEE Professional Communication Society International Professional Communication Conference and Proceedings of the 18th ACM International Conference on Computer Documentation: Technology & Teamwork, Cambridge, Massachusetts, September (2000) 497 – 508
2. Pearl Brereton: Component-Based System: A Classification of Issues. IEEE Computer, November (2000), 33(11): 54 – 62
3. Alan W. Brown: The Current State of CBSE. IEEE Software, September/October (1998) 37 – 46
4. Alan W. Brown: Large-Scale Component-Based Development. Prentice Hall, (2000)

5. Alan. W. Brown and Grady. Booch: Reusing Open-source Software and Practices: The Impact of Open-source on Commercial Vendors. Proceedings: Seventh International Conference on Software Reuse, Lecture Notes in Computer Science, Vol. 2319. Springer, (2002) 123 – 136.
6. Reidar Conradi, Tore Dybå: An Empirical Study on the Utility of Formal Routines to Transfer Knowledge and Experience. Proceedings of European Software Engineering Conference, Vienna, September (2001) 268 – 276
7. Bill Councill and George T. Heineman: Component-Base Software Engineering and the Issue of Trust. Proceedings of the 22nd International Conference on Software Engineering, Limerick, Ireland, June (2000) 661 – 664
8. Ivica Crnkovic and Magnus Larsson: A Case Study: Demands on Component-based Development. Proceedings of the 22nd International Conference on Software Engineering, Limerick, Ireland, June (2000) 21 – 31 .
9. Ivica Crnkovic: Component-based Software Engineering - New Challenges in Software Development. Proceedings of 25th International Conference on Information Technology Interfaces, Cavtat, Croatia, June (2003) 9 – 18
10. Torgeir Dingsøyr, Emil Røyrvik: An Empirical Study of an Informal Knowledge Repository in a Medium-Sized Software Consulting Company. Proceedings of 25th International Conference on Software Engineering, Portland, Oregon, USA, May (2003) 84 – 92
11. W. B. Frakes: An Empirical Framework for Software Reuse Research. Proceedings of the Third Annual Reuse Workshop, Syracuse University, Syracuse, N.Y. (1990)
12. W. B. Frakes, C.J. Fox: Sixteen Questions about Software Reuse. Communication of the ACM, June (1995), 38(6): 75 – 87
13. Ivar, Jacobson, Martin Griss, Patrick Jonsson: Software Reuse-Architecture, Process and Organization for Business Success. Addison Wesley Professional, (1997)
14. C.M. Judd, E.R. Smith, L.H. Kidder: Research Methods in Social Relations. Sixth edition, Holt Rinehart and Winston, (1991)
15. Y. Kim and E.A. Stohr: Software Reuse: Survey and Research Directions. Journal of Management Information System, (1998), 14(4): 113 – 147
16. C. Kruger: Software Reuse. ACM Computing Surveys, (1992), 24(2): 131 – 183
17. N. Y. Lee, C. R. Litecky: An Empirical Study on Software Reuse with Special Attention to Ada. IEEE Transactions on Software Engineering, September (1997), 23(9): 537 – 549
18. Marcus A. Rothenberger, Kevin J. Dooley and Uday R. Kulkarni: Strategies for Software Reuse: A Principal Component Analysis of Reuse Practices. IEEE Transactions on Software Engineering, September (2003), 29(9): 825 – 837
19. H. Mili, F. Mili, A. Mili: Reusing Software: Issues and Research Directions. IEEE Transactions on Software Engineering, June (1995), 21(6): 528 – 561
20. M. Morisio, C.B. Seaman, A. T. Parra, V.R. Basili, S.E. Kraft, S.E. Condon: Investigating and Improving a COTS-Based Software Development Process. Proceeding of 22nd International Conference on Software Engineering, Limerick, Ireland, June (2000) 31 – 40
21. Maurizio Morisio, Michel Ezran, Colin Tully: Success and Failure Factors in Software Reuse. IEEE Transactions on Software Engineering, April (2002), 28(4): 340 – 357
22. Parastoo Mohagheghi and Reidar Conradi: Experiences with Certification of Reusable Components in the GSN Project in Ericsson, Norway. Proceedings of the 4th ICSE Workshop on Component-Based Software Engineering: Component Certification and System Prediction. Toronto, May (2001) 27 – 31
23. Jeffrey S. Poulin: Measuring Software Reuse-Principles, Practices, and Economic Models. Addison-Wesley, (1997)

24. Vijay Sai: COTS Acquisition Evaluation Process: The Preacher's Practice. Proceedings of 2nd International Conference on COTS-based software systems, Lecture Notes in Computer Science, Vol. 2580. Springer, 2003, Ottawa, Canada, February (2003) 196 – 206
25. Guttorm Sindre, Reidar Conradi, and Even-Andre Karlsson: The REBOOT Approach to Software Reuse. Journal of System Software, (1995), 30(3): 201 – 212
26. Vu N. Tran, Dar-Biau Liu: Application of CBSE to Projects with Evolving Requirements- A Lesson-learned. Proceeding of the 6th Asia-Pacific Software Engineering Conference (APSEC' 99) Takamatsu, Japan, December (1999) 28 – 37
27. Padmal Vitharana: Risks and Challenges of Component-based Software Development. Communications of the ACM, August (2003), 46(8): 67 – 72
28. Katharine Whitehead: Component-Based Development: Principles and Planning for Business Systems. Addison-Wesley, (2002)
29. http://www.idi.ntnu.no/grupper/su/spike.html
30. http://www.ifi.uio.no/~isu/INCO/