ISSN 22773061

An Empirical Study on Software Reuse in Small IT Companies in the Balkan Region

[1st] Florinda Imeri1, [2nd] Ljupcho Antovski

State University of Tetovo, Ilindenska bb, 1200 Tetovo, Macedonia

florinda.imeri@gmail.com

University Ss. Cyril and Methodius, Rugjer Boshkovikj 16, 1000 Skopje, Macedonia

ljupcho.antovski@finki.ukim.mk

ABSTRACT

Software reuse as a distinct field of study in software engineering has been practiced since programming began. From different surveys it is seen that software reuse is an inevitable solution that has potential to improve time –to-market and man power/cost trends that have been ongoing, having a significant impact on software industry. It helps organize large-scale development and what is more important; it makes system building less expensive. In large companies many success stories of software reuse and use components have been quoted, with its potential for achieving good quality systems in short time scales but what about small, less structured companies, who depend on the ability to produce their product as quickly as possible, while trying to keep standards high enough to keep their customers happy and their maintenance costs low. One important issue is how to make best use of reuse at the companies of small size. In this article, we described the results of our survey at several IT companies in the region, mainly in Macedonia and Kosovo. Our aim was to evaluate issues surrounding software reuse and component-based development, from the perspective of developers involved in a software development. We wanted to explore their experience with software reuse and COTS components, to look at the possible benefits, disadvantages and contributors towards successful reuse and possibly try to increase the knowledge and understanding of CBSE.

Keywords: Empirical Study, CBSE, Software Reuse, survey, COTS, systematic reuse, reusable libraries, reuse metrics

Council for Innovative Research

Peer Review Research Publishing System

Journal: INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY

Vol 9, No 2

editor@cirworld.com
www.cirworld.com, member.cirworld.com

ISSN 22773061

INTRODUCTION

Today software has an enormous expansion use in everyday life, such as in business, industry, administration, research, etc. Traditionally, software development was challenged of increasing complexity. To satisfy delivery deadline and budget requirements they did not take into the consideration evolutionary needs of the system, which increased costs associated with software maintenance[1].

As enterprise software systems are continually growing in complexity, the main challenge for software developers is to adapt quickly to changes. To build new systems with little effort, IT industry is forced to find ways to streamline development process[2]. Over last decade's software development process is shifted, from scratch to assembling components developed by third party. This means that traditional approach should be changed to new design process based on systematic software reuse focusing on selection, evaluation, integration, and evolution of reusable software components. This new approach is attracting more and more attention from academia and practitioners as a solution with potential to improve time to market, increase quality and productivity and reduce work force. Many success stories were quoted from large companies for some time now, with its potential for achieving good quality systems in short time scales by the reuse of currently available components. But, what about the small, less structured companies, who depend on the ability to produce their product as quickly as possible, while, trying to keep standards high enough to keep their customers happy and their maintenance costs low[3].

Based on the research topics investigated, implemented using empirical data, we would like to evaluate issues surrounding software reuse processes in IT companies in region. This study interests to check whether companies promote reuse, which are factors that contribute toward reuse, what effect it has in the software development, do companies agree to use external components, which are advantages and problems that their use bring to the developers, and finally do they measure reuse benefits.

This paper is structured as following: Section 2 gives a literature overview on CBSE and software reuse, Section 3 in on similar works, Section 4 discusses research context. Further Section 5 has a short description on respondents, Section 6 contains research topics, Section 7 and Section 8 presents respectively discuss results of the survey while Section 9 concludes on results.

CBSE AND SOFTWARE REUSE

Component-Based Software Engineering (CBSE), as a new style of software development, involves all practices needed to build software systems with predictable properties[4]. It emphasizes reusability—that is, the creation and reuse of software building blocks [5], called software components. A software component is an implementation, in software, of some functionality, reused in different applications, and accessed via an application-programming interface. It may or may not be sold as a commercial product[4], it must be cataloged for easy reference, standardized for easy application, and validated for easy integration. According to Peter Freeman, reuse is the use of any information which a developer may need in the software creation process [6]. Reuse encompasses all life cycle assets such as requirements, design and code, through specific procedures of documentation. Usually reuse is done in an informal and non-systematic way. If this is done systematically, it would bring many benefits.

Software reuse can be specified in two directions [5], development for reuse and development with reuse. Development for reuse refers to systematic generalization of software components for later reuse, while development with reuse deals with how existing components can be reused in existing and new applications. When it comes to reusing in-house built components, these two processes are very much related.

According to Bennatan[7] software components can be categorized as:

- **COTS** (commercial off-the-shelf) components purchased from a third party, fully validated and are ready for use.
- **Off-the-shelf components** -software acquired from a third party or internally developed for a past project.
    - **Full-experience components**. Existing specifications, designs, code, or test data developed for past projects that are similar to the software to be built for the current project. Modifications required for full-experience components are at low-risk.
    - **Partial-experience components**. Similar to full experience components but requires substantial modification, which may have a fair degree of risk.
- **New components**. Components which are to be built by the software team specifically for the needs of the current project

SIMILAR WORK

There have been many researches in various perspectives showing that applying software reuse in software development is very challenging. It requires tremendous efforts and up-front investment to make the change. Despite several years of trying to bring reuse to practice, software engineers have found out that reuse in software is not the same as in other areas, that software is very hard to reuse[8]. The results of integration have lead to increasing managerial difficulties. But surveys say that large companies found that reuse works, and can pay off handsomely. There are significant corporate reuse programs at AT&T, HP, IBM, GTE, NEC, Toshiba, Hitachi, Ltd, Motorola, Inc., National Aeronautics and Space Administration (NASA).

A survey conducted by Lim[9], in Hewlett-Packard, with two reuse programs has show a substantial return on investment.

According to Griss[10] the Japanese approach to reuse concentrated on functionality, design, productivity, and quality showed a tremendous increase in productivity and quality. Thus Hitachi reduced the number of late projects from 72 percent in 1970 to 12 percent in 1984, and reduced the number of defects to 13 percent of the 1978 level. Toshiba increased productivity by a factor of 250 per- cent between 1976 and 1985, and by 1985 had reduced defect rates to 16 to 33 percent of the 1976 level.

A survey conducted at Nippon Telegraph & Telephone Corp. (NIT) showed they had a comprehensive program including a reuse-specific organization, printed catalogs, guidelines, and certification for reuse, leading to reuse levels of 15 percent or more, with several hundred small components[11].

In 1995, Frakes and Fox[12] conducted a survey at 29 organizations with a large number of respondents; 25,000.They asked 16 common questions on software reuse. Some of the results from the study revealed that programming languages do not affect to reuse, education and experience influence and that developers do not have the syndrome NIH (not invented here).

All these surveys are conducted in large companies. Such companies cannot be found in region. We are aware that our survey cannot be compared directly to them, but they serve as a guideline helping us to discuss the results obtained in our survey.

RESEARCH CONTEXT

The definition of “Small” and “Very Small” Companies is challengingly ambiguous. According to Johnson and Brodman[13] small companies are defined as companies with “fewer than 50 software developers and a small project as fewer than 20 software developers” whereas Laporte et al[14] have introduced a new definition for VS as “any IT services, organizations and projects with between 1 and 25 employees”.

According to Laporte[11], very small companies are economically vulnerable as they are driven by cash-flow and depend on project profits, so they need to perform the projects within the budget. They usually have low budget which have many impacts, such as: lack of funds to perform corrective post delivery maintenance; few resources allocated for training; little or no budget to perform quality assurance activities; no budget for software reuse processes; low budget to respond to risks; and Limited budget to perform Process Improvement and /or obtain a certification/assessment.

IT companies, respondents of our survey, have all attributes of very small companies; they all have less than 25 people involved in the process of software development.

This research relies upon collection of empirical data with an aim to explore developers’ experience on reuse and CBSE. In this case the focus is on two data gathering techniques: questionnaires and interviews. The questions of the questionnaire presented in the Table.1 are used for data collection from IT companies. The questionnaire contains 33 questions. Some of the questions were previously used at earlier studies on software reuse, but are adapted specifically for our purpose [12][15].

The questionnaire is designed with predefined answers and some open questions for general reflection. The questions try to address the issues identified in section 6. The questionnaires are done with 10 IT companies in the region, Macedonia and Kosovo. Interviews are done with four project leaders.

The data is both “quantitative” from questionnaire and “qualitative” from interviews with project leaders. The “quantitative” data is transformed into percentage representation while “qualitative” data from interviews is used as a guide for analysis.

We believe that this research method supplemented with data material enables us to get satisfactory information.

RESPONDENTS

Respondents are mainly software development companies, whose application domain is information systems such as administrative, commerce and educational software. 8 companies are from Macedonia whereas two are from Kosovo. We had contacted project managers beforehand to get their agreement, and made sure they will have enough time to complete it. A questionnaire was sent electronically. Most of respondents have bachelors or masters in computer sciences and 60% of respondents have more than 5 years of working experience. They had one week to complete the questionnaire. In addition to the data collected from the questionnaire, project leaders were interviewed.

RESEARCH TOPICS

Following are presented research topics of our survey. The questionnaire and the table which show the relation of the research topics to the questions of the questionnaire is presented as well.

RT1. Do companies develop from scratch or reuse?

Since 1969, when for the first time at the NATO conference was discussed and proposed the term Software Engineering there has been a remarkable progress in software development process. Yet, there are still problems with software projects. Most of conventional methodologies such as object-oriented methodology assume development from scratch. Software reuse is the primary source of productivity and quality [16]. Although object-oriented technologies have promoted software reuse, there is a big gap between the whole systems and classes. To achieve significant payoffs a reuse

program must be systematic. There are two types of software reuse: **Opportunistic reuse** – when starting a project, developers may find there are existing components that can be reused directly and **Planned or systematic reuse**- developers strategically design components which are to be reused in future projects.

RT2. Which factors may influence toward systematic reuse?

Introducing reuse processes and modifying non-reuse processes is a key point for success, for which top management commitment is a pre-requisite[17]. To apply systematic reuse developers need to have knowledge and information on a per component basis [18]. Since most software engineers are not educated about software reuse, it is up to the organizations to train them.

Reuse depends on **technical factors**: programming language support, repositories and tools for searching reusable artifacts and **non-technical factors**: organization, process and human involvement.

RT3. Do companies use COTS and does this increase rework?

COTS, third-party components, is a prebuilt software that is integrated into the system supplied by e vendor, whose internals cannot be viewed or accessed leaving us with their externally observable elements for evaluating their quality characteristics. Gartner study noted that companies that adopt component-based development see a 50% improvement in programming cost[19]. By using this approach they expect either more rapid or less costly system construction as they hope to stay in step with the technological advancements happening in the competitive marketplace. There are disadvantages of COTS which needs to be considered [20], such as finding, understanding and adapting, increased maintenance costs, creating and maintaining a component library. The other risk is that a purchased component may turn out incapable, or may not perform as well as expected.

RT4. Does reuse payback?

Reuse has proven to be a long term investment showing benefits over a series of projects often spanning several years [21]. If a short term payback is expected, than disappointing early results can cause a program to be canceled before it has a chance to pay off. To determine if reuse program is succeeding, the measurement of reuse is crucial.

Table 1. The questionnaire

Q1.	Which is your application area of your institution?
Q2.	What kind of application domain/software is developed by your organization?
Q3.	How many employees from your organizations are involved with software development activities?
Q4.	What is the average experience of your organization's development team?
Q5.	Your level of education?
Q6.	Which development approach is used by your organization? (object-oriented or component-based)
Q7.	How is the software production process carried out in your organization?(product families or isolated products)
Q8.	Do you consider re-usability as an important software quality when you start a project?
Q9.	When a development team set up a project. Would they start to go through old code or software, or start to write the new program instantly?
Q10.	Which is the percentage of code reuse?
Q11.	Your experience with software reuse?
Q12.	In your organization, is there any software reuse education program? If yes have you attended?
Q13.	Does your organization have any systematic reuse process?
Q14.	In your organization, is there a repository for storing and retrieving reusable assets? Is it beneficial to you?
Q15.	In your organization are there any forms of incentive or recognition to increase software reuse?
Q16.	Is there a position called "reuse engineer" in a development team/organization based on your experience?
Q17.	Assets you mainly use are internal or external?
Q18.	Does management agree using COTS?

ISSN 22773061

Q19.	How do you buy components?
Q20.	Were these components small or complex?
Q21.	Which you think are benefits to using COTS?
Q22.	How difficult is to search and find COTS?
Q23.	What problems you faced while searching COTS?
Q24.	What kind of process is followed when planned to use COTS?
Q25.	Who does the integration part?
Q26.	Which are the most common problems with the glue code?
Q27.	Do you have problems with the COTS documentation?
Q28.	Do COTS affect to the requirements?
Q29.	From your experience does COTS damage/modify the existing architecture?
Q30.	Does COTS affect the maintenance?
Q31.	Have your organization obtained success in projects through software reuse?
Q32.	Have you ever measured Reuse? How much time/budget saved?

Table 2. Research topics vs. questionnaire questions

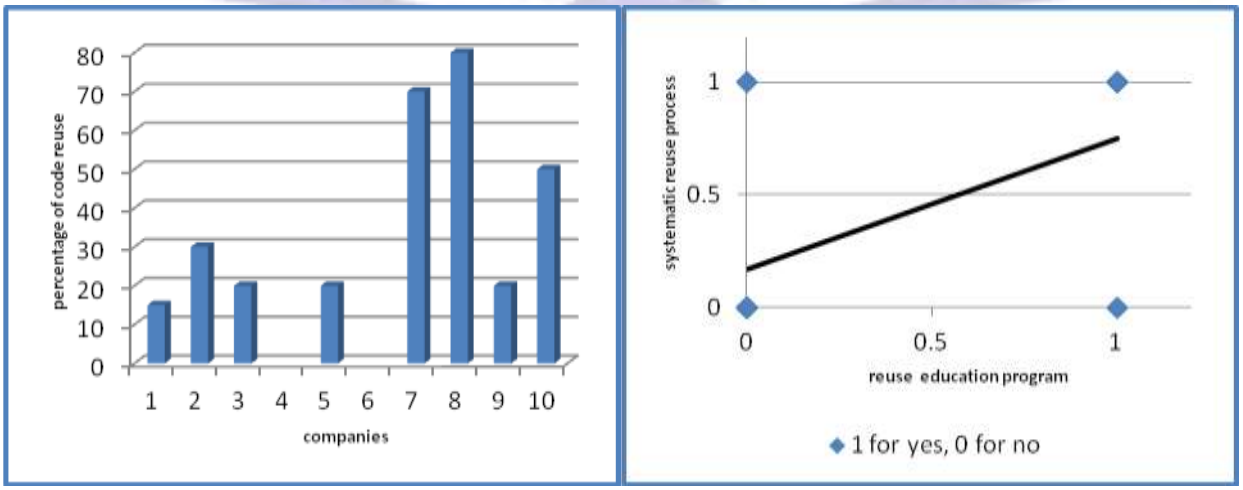
Research topic	RT1	RT2	RT3	RT4
Questions	Q6-Q10	Q11- Q16	Q17-Q30	Q31, Q32

PRESENTATION OF RESULTS

Following, results of our survey are summarized. The statistical data presented in this study are based on answers, which are relevance to our research topics. We used Microsoft Excel 2007 as statistical analysis tool.

RT1. Do companies develop from scratch or reuse?

Answers to questions Q6-Q10 gave us information about the way companies develop software, whether they use a formal software reuse process or start from scratch. Regarding Q6 we found that 80% use object-oriented approach while 20% uses both, object-oriented and component-based approach. For Q7 we found that 40% develop products that evolve over time, 30% develop isolated products, specific software for each project, and 30% develops both kind of products. Further Q8, all respondents, 100%, see re-usability as an important software quality when starting a project. Q9, 90% of companies start through old code or software, checking what can be reused, 10% start from scratch. Finally Q10, the percentage of code reused, is shown in Fig.1.



ISSN 22773061

Figure 1. Percentage of code reused systematic reuse

Figure 2. Reuse education program vs.

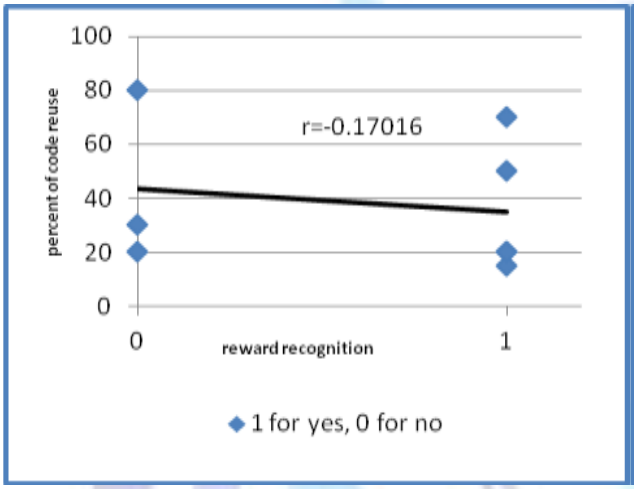
RT2. Which factors influence toward systematic reuse?

The questions which are used to investigate this research topic were questions Q11-Q16. From interviews with project leaders we know that management supports software reuse. Results from Q11 show that 80% of developers are informed for software reuse from work whereas 20% from education. Question Q12, shows that 40% of companies have software reuse education program whereas 60% don't have. Regarding process development, question Q13, 40% of companies perform reuse systematically, 50% do not, and one company did not answer the question.

To see whether enterprise reuse education program influences the systematic reuse, we compare results of Q12 and Q13, Results show a reasonable positive correlation (r=0.58) among reuse education program and systematic reuse Fig.2.

We wanted to see whether companies have their own repository for storing and retrieving assets, and from the answers to question Q14, it resulted that all ten companies have their own repository and they find it beneficial. Q16 shows that none of companies have any position called "reuse engineer".

Reward incentives may be catalysts for reuse, question Q15. There are two kinds of reward, by cash and by recognition. 60% answered yes; there are rewards, while 40% said no. To check whether reward increases reuse we compared it with percentage of code reuse, Fig.3. Results show a low negative correlation (r=-017) among reward and percentage of code reuse.



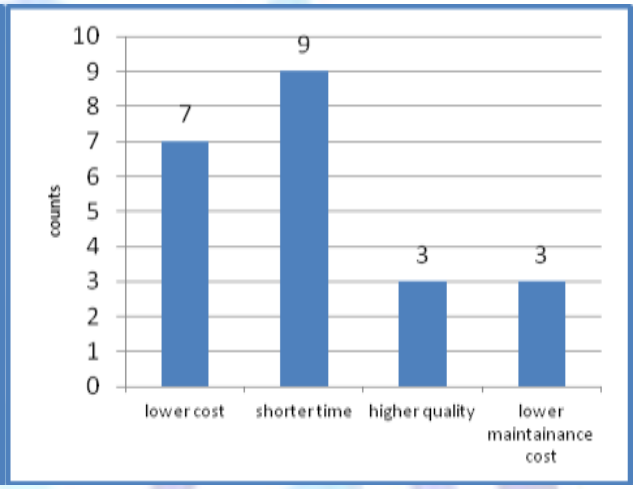


Figure 3. Reward recognition vs. percentage of code reuse

Figure 4. COTS benefits

RT3. Do companies use COTS and do they increase rework?

Questions used to evaluate this research topic are questions Q17-Q30. From Q17, we see that 90% of companies mainly use in-house (internal) assets while 10% uses both.

Question Q18, all 10 companies are in favor of using COTS. Regarding Q19, 80% said they buy white-box COTS and 20% black-box COTS. Whether these COTS were small or complex, question Q20, 30% said they were small, 70% they were complex and 10% both. Answers for Q21, benefits to using COTS from their perspective, are presented in the Fig.4. 60% of companies declared they find partly difficult to search and find COTS whereas 40% finds it easy, question Q22. None of the respondents answered to the question Q23. Further Q24, 70% of companies study available components, it means they trust the suppliers, 30% study both, suppliers and components. Regarding integration, Q25, we had different answers, such as team leader, programmer and a specific team.

In Fig.5 we present the data we collected regarding the problems that COTS brings to the team, questions Q26-Q30, such as problems they have with glue code abbreviation **wgc**(write glue code), **tgc**(test glue code); problems with documentation, abbreviation **doc**; changes to the requirements and architecture, abbreviation **req** and **arch**; problems with maintenance, abbreviation **main.**

RT4. Does reuse payback?

And finally from their experience we wanted to see whether reuse payback. Questions Q31 and Q32 will help use evaluate this research topic. First, we asked whether they measured reuse, Q32, and how does it affect to their budget. 90% responded that they cannot measure reuse since they find it difficult and only 10% said yes. And finally, Q31, from their experience, 60% responded yes, and have mention projects which resulted successful thanks to reuse while 40% responded no.

ISSN 22773061

DISCUSSION OF RESULTS

Based on the results obtained from our survey we will discuss our research topics.

RT1. Do companies develop from scratch or reuse?

The results from the survey provided very valuable insights into the attitudes and working practices of the company's staff. Typically, projects are built as "one-time only," without reuse in mind, and tend to be tightly bound within themselves.

From the results we can conclude that developers prefer to reuse rather than build from scratch, but their reuse is ad-hoc-opportunistic. There are several indications to this such as companies developing processes, method they use, version of their products they build. They feel that packing assets takes time, since usually there is lack of time because of pressure from the market.

We believe that management is responsible to introduce and convince developers that reuse is economically viable practice.

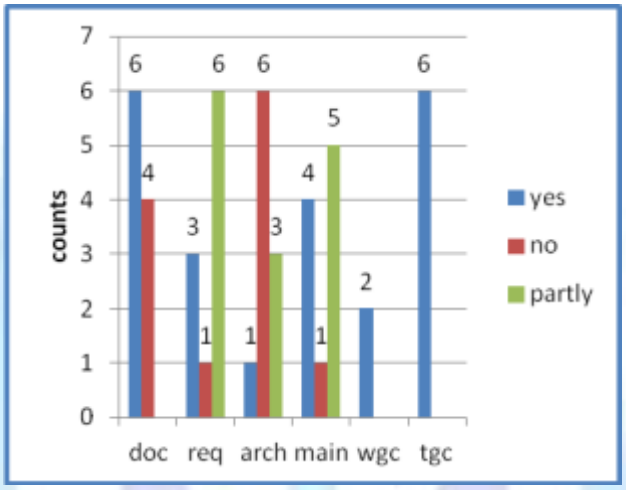


Figure 5. COTS disadvantages

RT2. Which factors influence toward systematic reuse?

Frakes &Fox [12] in their paper tried to find factors which facilitate reuse. They found that reuse education does influence. Based on the information we got from the questionnaire, most of the developers are not educated about software reuse in school. What we have seen is that a company with corporate reuse education program performs reuse systematically, meaning that training/education influences reuse, as seen from the Fig.2. This conclusion supports the significance of reuse education program.

Libraries of reusable code and other assets are important, but they will not be fully utilized without process support of reuse. In accordance to this, we found that owing a repository does not influence toward systematic reuse. We believe that a group which will be responsible for the reuse infrastructure may influence.

We also wanted to see whether reward recognition in any form will increase reuse level. As in [12] we compared percentage of code reuse, with responses to rewards recognition. We found that reward does not make a significant difference (Fig.3).

RT3. Do companies use COTS and does this increase rework?

Based on the interviews we had with project leaders, we have seen that all companies in principle agree using components from third parties. We believe that companies do not suffer from the syndrome NIH (not invented here), but in practice they mostly use internal assets. When it come to the COTS benefits Fig. 4, in accordance to previous surveys, most of the companies think that short time to market and lower costs are seen as equally important benefits whereas high quality and short maintenance cost are less important. We found that developers are unhappy with documentation. They have problem with integrating them since most of them have problems with glue code. Even though literature suggests that trusting to the suppliers is an issue, companies did not see this as a problem, because they know suppliers.

We have seen no significant indication that COTS increases rework, since they mainly buy them with source code which makes it easier to modify and adapt them.

RT4. Does reuse payback?

All companies agree that reuse payback, and they also mention by name projects which were successful thank to reuse, but the lack of reuse metrics in almost all companies makes it impossible to measure the effects that reuse has in their budget.

CONCLUSION AND FUTURE WORK

To keep up with market trends, small companies unlike larger companies need to succeed in every project they undertake. They cannot afford to fail in any project, because the livelihood of every employee and the company depends upon keeping their market share. If products fail, then the company would cease to exist. The willingness to take unexplored risks is much smaller since stakes are much higher in a small company.

This study has been very useful to view the workings of a small company under pressure to meet customers' requests and demands for new products. In general the empirical data collected in this study represents an overview of the current situation among companies regarding reuse processes. The most evident patterns that could be identified from the data can be summarized as follows.

Companies in general are in favor of implementing reuse processes. This was also suggested by developers' in the questionnaire where 100% of them agreed that reuse is an important software quality. Introducing systematic reuse will inspire staff to improve their development ideas and processes. We believe that organizational improvements are required for the successful implementation of a reuse program. It requires resources for an appropriate training with developers and maintenance. Development of a reuse process and repository produces a base of knowledge that improves quality after every reuse, minimizing the amount of development work required for future projects, and ultimately reducing the risk of new projects that are based on repository knowledge. In order to build quality reusable software and achieve the most of reuse, standard coding practices and code documentation must exist across the organization.

Commitment of management and staff to reuse can make a significant difference to both productivity and profitability.

As organizations implement systematic software reuse programs to improve productivity and quality, they must be able to measure their progress in order to identify the most effective reuse strategies. This is done with reuse metrics and models. So for future research it will be of interest to have more extensive feedback from companies on the relationship between reuse, quality and productivity.

ACKNOWLEDGMENTS

This work is done as a part of our PhD research. Its focus is to analyze the effects that software reuse and software components has on the quality of information systems developed in region. We would like to thank all the companies for giving us the opportunity to collect data.

REFERENCES

[1] I. Crnkovic, S. Larsson, and J. Stafford, "Component-Based Software Engineering : Building systems from Components," 9th IEEE conference on CBS, 2002.

[2] Syed Raza Kirk Knoernschild, "Software reuse for business success," developerFusion. 2010.

[3] P. Gibs, "A Case Study in Small Company software reuse," University of Durham, 1999.

[4] C. Buhman, F. Long, and R. Seacord, "Volume I : Market Assessment of Component-Based Software Engineering," Internal Research and Development, vol. I, p. 43, 2001.

[5] M. Ramachandran, "Software reuse guidelines," IRI -2005 IEEE International Conference on Information Reuse and Integration, vol. 30, no. 3, pp. 1–8, 2005.

[6] M. Ezran, M. Morisio, and C. Tully, Practical software reuse. 2002.

[7] E. M. Bennatan, Software Project Management: A Practitioner's Approach, vol. 2. McGraw-Hill Publishing Co, 1994, p. 237.

[8] R. Prieto-Díaz, "Reuse in Engineering vs. Reuse in Software: Why Are They Incom-patible," Symposium on Software Reusability SSR, 2001.

[9] W. Lim, "Effects of reuse on quality, productivity, and economics," Software, IEEE, 1994.

[10] M. L. Griss, "Software reuse: From library to factory," IBM Systems Journal, vol. 32, no. 4, pp. 548–566, 1993.

[11] M. Aoyama, "CBSE in Japan and Asia," Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2001, pp. 213–225.

[12] W. Frakes and C. Fox, "Sixteen questions about software reuse," Communications of the ACM, vol. 38, no. 6, pp. 75–87, 1995.

[13] J. G. Brodman and D. L. Johnson, "What small businesses and small organizations say about the CMM," in 16th International Conference on Software Engineering, 1994, no. July 1991, pp. 331–340.

[14] C. Laporte, A. April, and A. Renault, "Applying ISO/IEC software engineering standards in small settings: historical perspectives and initial achievements," in Proceedings of SPICE Conference, 2006, pp. 1–6.

[15] O. P. N. Slyngstad, A. Gupta, R. Conradi, P. Mohagheghi, H. Rønneberg, and E. Landre, "An Empirical Study of Developers Views on Software Reuse in Statoil ASA," Science And Technology, no. Idi, 2006.

[16] C. Szyperski, D. Gruntz, and S. Murer, Component Software Beyond Object-Oriented Programming, vol. 2E. 2002, p. 626.

[17] N. Fenton and M. Neil, “Software metrics: successes, failures and new directions,” Journal of Systems and Software, pp. 1–19, 1999.

[18] I. Crnkovic, “Component-based software engineering—new challenges in software development,” Software Focus, vol. 2, no. 4, pp. 127–133, 2001.

[19] J. Williams, “Distributed Components and the Internet,” Application Development Trends, pp. 79–80, 1999.

[20] I. Somerville, Software Engineering, 8th ed., vol. 8. Addison-Wesley Publishers Limited, Pearson Education 2007., 2006.

[21] J. E. Gaffney and R. Cruickshank, “A general economics model of software reuse,” ICSE ’92 Proceedings of the 14th international conference on Software engineering, pp. 327 – 337, 1992.

AUTHORS’ BIOGRAPHY

Florinda Imeri, Mr. Sc is a lecturer at the State Univeristy of Tetovo, Faculty of Mathematical Sciences, Computer Science Department, Tetovo, Republic of Macedonia. She has completed her Mr. Sc in Computer Science in 2008. She is doing her PhD in Software Engineering. Florinda Imeri is engaged in several research areas: Software Engineering, Software Components, Software Project Managements, Operating Systems, Object Oriented Programming.

Ljupcho Antovski, PhD, is associate professor of software engineering at the Faculty of Computer Science and Engineering, St. Cyril and Methodius University, Skopje, Macedonia. He lectures courses in requirements engineering, software project management, e-business, and mobile applications. He is the president of the IT technical committee at the Macedonian Institute for Standardization. Ljupcho is cofounder and board member of iVote, leading integrator of elections management information systems in South-Eastern Europe. He has vast experience in implementing election information systems worldwide. His academic research is focused on the requirements and usability engineering when implementing information systems, and the broad applicability of mobile voting. Ljupcho has published widely in the fields of software engineering, m-government and m-voting. He has authored books and numerous papers in leading journals and conference proceedings.