

Requirements Reuse: the State of the Practice

Yuri Chernak, Valley Forge Consulting, Inc

Abstract – For several decades, software reuse has been a recognized solution to improving efficiency of software development. However, implementing reuse in practice remains challenging and the IT community has little visibility into the state of the practice specifically as it pertains to reusing software requirements. This paper presents the results of a survey conducted in the global IT industry in 2010 and discusses the state of the practice for software requirements reuse. The survey studies reuse adoption in two different contexts, i.e., Software Product Lines and Software Maintenance. The analysis of the survey data focuses on the latter context as a more common case in practice and investigates the impact of various factors on reuse adoption and effectiveness.

Index Terms – Change impact analysis, requirements engineering, requirements reuse, software maintenance, software product lines



1 INTRODUCTION

Software reuse is a recognized solution to the three primary drivers of the software business – **faster, better, cheaper**. This topic has been the focus of the IT industry for several decades and it has been addressed in many publications, e.g. [1- 5]. All major products of a software process, for example requirements, architectural components, and software code, can be subject to reusing in two separate contexts – a) for developing similar applications, and b) for implementing releases of the same application. The first context is known as Software Product Lines (SPL), a.k.a., Domain Engineering [6 - 8]. The second context is known as Software Maintenance [9 - 11].

In practice, requirements reuse for software maintenance appears to have much more opportunities as opposed to the SPL case for two reasons:

- Firstly, software maintenance is a more common case in practice as most of the applications in a typical IT department, for example in the financial or hospitality industries, are existing production systems that go through regular cycles of production releases. Whereas, new applications account only for a small share of the IT portfolio. This fact leaves little opportunity for IT project teams to apply and benefit from requirements reuse in the SPL context.
- Secondly, most of the time changes allocated to future releases overlap with the existing functionality of applications. Hence, this provides opportunities for reusing the existing product requirements for future releases.

Broad adoption of requirements reuse in the IT industry still remains challenging for both contexts. So far, IT research has been primarily focused on studying requirements reuse in the SPL context [12, 13], whereas requirements reuse for releases in the Software Maintenance context has had much less attention. As a result, the IT community has little visibility into the state of the reuse practice in the latter context. Nor are the obstacles for adopting reuse and factors impacting reuse effectiveness well-understood. Lastly, IT practitioners lack recommendations for better reuse adoption for releases.

This article discusses the state of the practice of software requirements reuse in the global IT industry and presents the results of the survey conducted between March and August, 2010. The goal of this survey was twofold:

- a. Gain visibility into the state of the practice of software requirements reuse in the global IT industry with a focus on the Software Maintenance case;
- b. Identify common obstacles for adopting requirements reuse as well as the factors impacting reuse effectiveness; provide recommends to practitioners to better adopt reuse.

2 SURVEY DESIGN AND DATA COLLECTION

The survey was intended to collect data from the global IT community. To accomplish this goal, a web-based survey was developed using the SurveyGizmo™ tool (www.surveygizmo.com), which specializes in developing online surveys, and collecting and managing survey data. The survey included 22 questions structured into four pages:

- **Welcome Page.** This page explained the purpose of the survey, as well as who will analyze the survey results and how they will be communicated.
- **Tell Us About Requirements Reuse.** This page included ten questions about a respondent's experience with requirements reuse on the latest project.
- **Tell Us About Your Latest Project.** This page included six questions for participants about their latest project characteristics, approach to developing requirements, etc.
- **Tell Us About Yourself and Your Company.** This page also included six questions about each participant's IT experience, role on a project, and basic information about his or her company.

The key to the survey's success was in getting a sufficient number of responses. The survey invitations were announces within various professional email groups, posted on a few IT-related websites on the Internet and also sent to many individuals in the author's professional network. The survey was conducted for six months, from March to August, 2010. The total number of individuals who opened the survey was 199, and among them 82 participants completed the survey and submitted their responses.

3 PROFILE OF SURVEY PARTICIPANTS

As the survey was conducted online, IT professionals all over the globe were able to participate. Figure 1 shows the allocation of participants on the world map from which we can see that responses came from all continents, except for South/Central America.

Figure 2 shows the distribution of participants' organizations by business sector. The chart shows that the biggest contributors to the survey were participants from the Financial/Banking and Technology sectors – 35% and 23% respectively, followed by the Government sector (13%). The other sectors combined represented 29%.

The biggest share of responses (46%) came from the mid-size teams of 6-10 people and then followed by the large teams of 21-50 people who represented 21%. The smallest teams of 1-5 people represented only 10% of responses.

Over 50% of participants were acting business analysts (BA). Other participants had various IT roles, but all had BA experience in the past. Finally, most participants (67%) had significant, i.e., over 5 years of BA experience.



Figure 1 Global view of survey participant locations

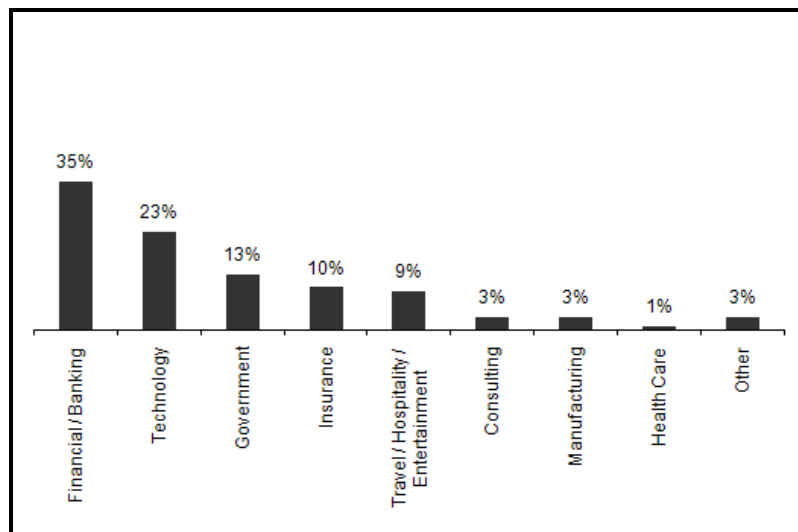


Figure 2 Distribution of Responses by Company Sector

4 Reuse Importance, Benefits, and Obstacles

The survey questionnaire began with asking the participant's personal opinion of whether requirements reuse is important and can provide benefits. Most of the survey participants (80%) believe that requirements reuse is important and can provide benefits for the project teams. Only a small number of participants (7%) believe that requirements reuse is not important. Some participants (13%) believe that requirements reuse is important but that its benefits are conditional and depend on other factors.

To capture the input about the "It Depends" factors the survey provided a free-text field. This information can be summarized as a list of four factors impacting reuse benefits:

- Requirements maintenance cost
- Similarity of applications
- Existing requirements quality
- Existing requirements structure and the level of abstraction

A common view in the IT industry on the benefits from reusing software artifacts, e.g., architectural components or source code, is that it improves time-to-market, the quality of products, and reduces the cost of development. One specific research objective of the survey was to understand whether the benefits of requirements reuse are consistent with the benefits reported from reusing other software project artifacts. The survey collected inputs from practitioners about what benefits they expected or actually achieved from reusing requirements where the most common response was "faster time-to-market" followed by the "lower development cost" response. This survey finding indicates the consistency between the benefits from reusing requirements with the benefits from reusing other software artifacts.

The next survey question was about the participant's latest reuse experience. The survey found that, despite the fact that most participants believe that requirements reuse is important and beneficial,

only 59% of respondents actually reused requirements on their latest projects. In anticipation of the fact that many participants may not reuse requirements, the survey included a question about the obstacles for adopting requirements reuse and provided six options to answer this question:

1. The project team did not feel that reuse is important and worth the effort
2. The project management did not support requirements reuse
3. The requirements developed in previous releases were incomplete (or do not exist), so it is impossible to reuse them
4. The existing requirements were poorly structured, so it is difficult to identify which requirements can be reused
5. The existing requirements are not kept updated, which makes them difficult to reuse
6. Other reasons

When answering this question, respondents could select multiple options. Obstacles 3, 4, 5 from the above list were the top three most commonly reported and they all relate to the quality of existing requirements. This means that to be reused, requirements should be complete, structured and maintained; only then can reuse be feasible and bring benefits. This is consistent with the “It Depends” factors discussed above.

5 Reuse Adoption Factors

A number of factors related to project specifics and the requirements process can impact the adoption of reuse. From this perspective, the survey investigated:

- In which of the two contexts (SPL or Software Maintenance) IT teams more commonly adopt and practice requirements reuse
- Which schools of developing requirements are more common in the IT industry
- How the rate of reuse adoption can be different among various schools

5.1 Two Reuse Contexts

There are two different contexts for discussing and implementing reuse of software requirements – a) Software Product Lines, i.e., reuse for a family of similar products, and b) Software Maintenance, i.e., reuse for releases of the same application. The former case has been the focus of research in the IT community for over two decades, whereas the latter case lacked attention, specifically from the requirements reuse perspective.

Commonly, most business applications in a typical IT department are existing production systems that go through regular cycles of production releases. Whereas, developing new applications accounts for a small share of an IT portfolio. This suggests that the Software Maintenance context can be a much more common case in practice and it can provide more reuse opportunities as opposed to the SPL case. Hence, one of the survey objectives was to collect data and provide visibility into which of the two contexts of requirements reuse is more common.

83% of respondents, who practiced reuse, reported that they reuse requirements within the context of releases of the same application; this finding confirms that the case of Software Maintenance is a much more common reuse context. Given that adopting reuse in this context remains challenging, researchers in the IT industry should study this case more closely and provide practitioners with solutions to better adopt requirements reuse and make it effective.

5.2 Approach to Developing Requirements

Currently, there are three major schools of developing and documenting requirements in the IT industry – Traditional Requirements Engineering (RE) [14, 15], Use-Case-Driven [16, 17], and Agile [18, 19] methodologies. As shown in Figure 3 most of the survey respondents (48%) follow the Traditional RE approach; 28% of respondents follow the Use-Case-Driven approach. Whereas, only 11% of respondents reported that they follow the Agile approach. The approaches that did not fall in either of these categories were qualified as “Other” and such responses represented 13%.

The next factor to investigate was the dependency of the rate of reuse adoption on different schools of developing requirements. It appears (see Figure 4) that the project teams that follow the Traditional RE and Use-Case-Driven approaches reported much higher reuse adoption rates, 61% and 56% respectively, as opposed to the Agile teams that reported a much lower rate of 33%. This survey finding suggests an opportunity for improving the Agile methodology.

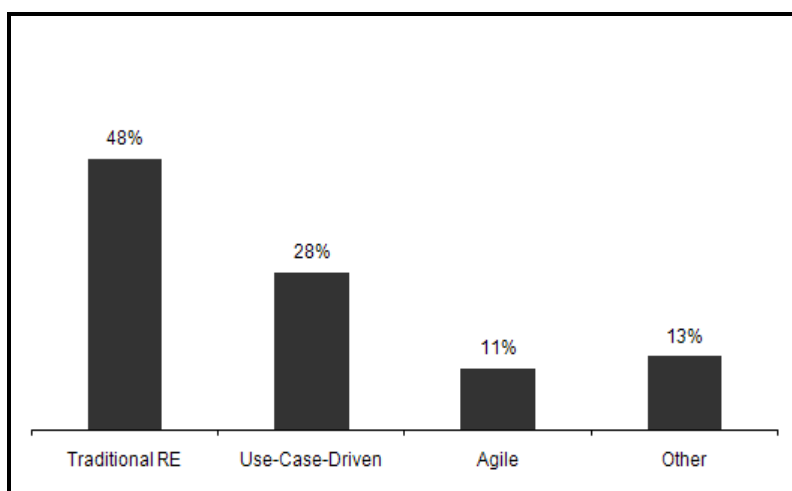


Figure 3 Distribution of Approaches to Developing Requirements

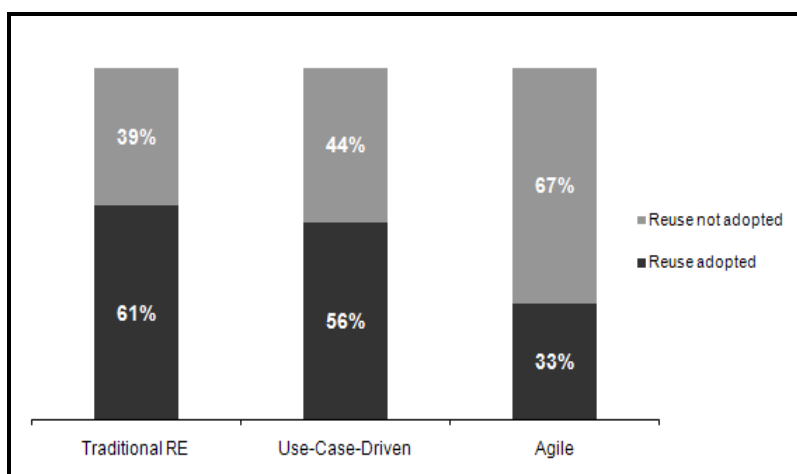


Figure 4 Reuse Adoption Rates as a Function of Approaches

6 Reuse Effectiveness Factors

When practicing requirements reuse, the project team should focus on demonstrating that the reuse process is effective and adds value. The main purpose of reuse is to help project teams save the time and effort to develop requirements, hence, the reuse benefits and reuse effectiveness will be in proportion to the reuse rate, which can be defined and measured as follows:

Reuse Rate is the ratio between the number of existing requirements reused from previous releases and the total number of requirements used to implement a given release.

For example, if a team follows an Agile approach, the reuse rate for a release will be calculated as a ratio between the existing user stories to be modified for reuse in the release and the total number of user stories necessary to implement the release. In this case, the total number of user stories will be the sum of the reused stories and new user stories developed for the release. In this article, the terms *reuse effectiveness* and *reuse rate* are used interchangeably.

The survey data analysis focused on reuse for releases of the same application and investigated reuse effectiveness in this context from the following perspectives:

- Understand the average reuse rate reported by practitioners and see the rates of the high-end of reuse effectiveness achievable in practice
- Investigate whether the approach to developing requirements can have a material impact on reuse effectiveness
- Investigate how practitioners decide which requirements can be reused for future releases, in particular, whether performing change impact analysis can help them achieve better reuse effectiveness

6.1 Reuse Rate as a Function of Approach to Developing Requirements

The survey found that the average reported reuse rate was 45%, and a number of respondents reported much higher reuse rates. In particular, 16% of respondents reported reuse rates in the range of 80 - 100%. This suggests that implementing a highly effective process to reuse requirements is achievable in practice.

The survey investigated whether the requirements methodology can have a material impact on reuse rates. The survey found that the highest average reuse rate of 64% was reported by the project teams that practice the Use-Case-Driven approach, followed by the teams who practice the Traditional RE approach (reuse rate of 43%). The lowest average reuse rate of 30% was reported by the Agile teams. This survey finding suggests that the Use-Case-Driven approach leads to significantly higher reuse rates and can be recommended as a primary option for teams who plan to adopt requirements reuse.

6.2 Reuse Rate as a Function of Separation Between Stakeholder and Product Requirements

The Requirements Engineering discipline defines a classification of requirements, where the following two categories of requirements [20, 21] are investigated in the survey:

- a. **Stakeholder** (a.k.a. customer, user, business) requirements. Stakeholder requirements are statements of the needs of a particular stakeholder or class of stakeholders; they are identified through *requirements elicitation*. This type of requirements is used to define and agree on a release scope, derive product requirements to implement the release, etc.
- b. **Product** (a.k.a. functional, software, solution) requirements. Product requirements are refinements of the stakeholder requirements; they describe the characteristics of a solution that meets stakeholder requirements. Product requirements are developed and defined through *requirements analysis*, and they are frequently divided into sub-categories, for example, functional, non-functional, supplementary, GUI, system interface, etc.

From the requirements reuse perspective the distinction between these types of requirements is very important and can help better answer two reuse-related questions:

- **Which category of requirements should be reused?** Between the two categories, only the product requirements, representing the application features, are subject to reusing for releases. In contrast, the stakeholder requirements, representing end-user needs, are not reused for releases. Once they have been implemented and satisfied in a given release, they are no longer needed for the next release. Further, the separation between the stakeholder and product types of requirements can help us better perform change impact analysis and better answer the next question.
- **How do we know which of the product requirements can be reused for a release?** The key to answering this question is performing change impact analysis. The purpose of this analysis is to

understand how the changes requested by the end-users impact existing functionality. Once the impact is analyzed and understood, decisions can be made about which of the existing product requirements are impacted by the requested changes and, therefore, can be modified and reused to implement the next release. This concept of reusing product requirements for releases is similar to the concept of maintaining and reusing software source code.

In practice, project teams frequently do not differentiate between these categories of requirements. As a result, it can be difficult to answer the above questions and reuse effectiveness may be impacted and reduced. Further, the classification of requirements into these two types is generic and not specific to any of the requirements development methodologies. However, the importance of this classification may not be addressed and followed by practitioners equally within the different methodologies.

The survey confirmed that separating the stakeholder and product types of requirements is not always the case in practice. In fact, many teams (49%) do not differentiate between these two types that negatively impacted the reuse effectiveness on their projects. The average reuse rate they reported was significantly lower (35%), as opposed to the average rate (54%) reported by the teams that follow the separation.

Next, the survey found that the concept of separating the stakeholder and product types of requirements is not equally followed within different schools of requirements. Figure 5 shows that separation was much more commonly practiced by the teams implementing the Use-Case-Driven approach (72%), then followed by the Traditional RE (47%) and Agile (43%) approaches.

As discussed above, change impact analysis is a critical activity when implementing requirements reuse for releases. Performing change impact analysis is a part of the general requirements engineering discipline [21]; guidelines and steps to perform this task can be found, for example, in [15, 22 - 24]. Thus, one of the survey objectives was to investigate how practitioners identify requirements for reuse and whether they perform change impact analysis for this purpose. Another objective was to understand if change impact analysis is performed, how it can help the team achieve better reuse effectiveness.

Figure 6 shows the distribution of responses to the question about how practitioners identify requirements for reuse. 69% of them did not answer this question, 15% of respondents indicated that this step is performed informally, 10% of respondents used change impact analysis, and 6% of respondents indicated that they used a requirements traceability matrix. This result indicates that most practitioners do not use change impact analysis and may not even be familiar with this technique; however, it is a critical activity for achieving an effective reuse process. As the survey found, the teams performing change impact analysis reported much higher reuse rates (on average 85%) as opposed to the teams who informally identify requirements for reuse, their reported reuse rates were significantly lower (on average 39%).

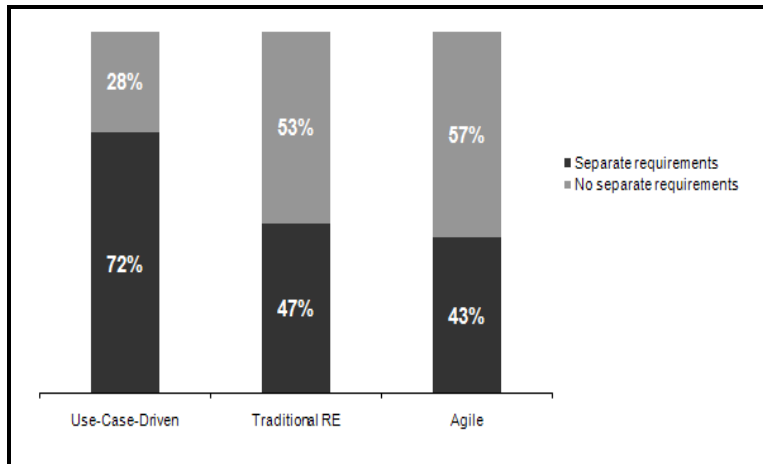


Figure 5 Separation Adoption as a Function of Different Approaches

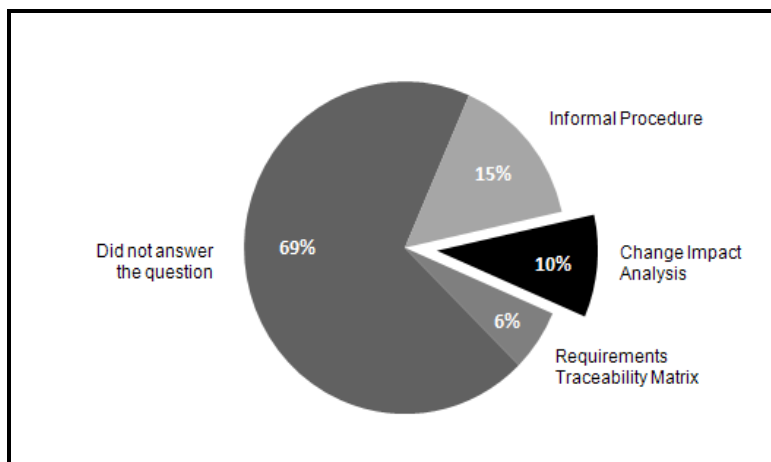


Figure 6 Techniques to Identify Requirements for Reuse

7 SURVEY SUMMARY AND CONCLUSION

This article presents the results of the survey conducted in the global IT industry between March and August, 2010. The survey goal was to investigate the state of the practice with requirements reuse, specifically focusing on the Software Maintenance case where requirements are reused for releases of the same application.

- Most of the survey participants (93%) believe that reusing requirements is important and can provide benefits for project teams. However, implementing reuse remains challenging and only half of the survey participants actually practiced reuse. Researchers in the field of requirements engineering should further investigate the factors leading to better adoption of requirements reuse and provide recommendations to practitioners.
- The survey confirmed that the main benefits of requirements reuse are reductions in time-to-market and development cost, which is consistent with the benefits from reusing other software artifacts.
- The main obstacle reported for adopting requirements reuse is poor quality of existing requirements. Having unstructured, incomplete, outdated existing requirements makes it difficult

to reuse them going forward. Developing techniques to analyze the inventory of and refactor existing requirements can help practitioners better adopt and benefit from reuse.

- Between the two reuse cases – software product lines vs. software maintenance (releases of the same application), the latter case is most common in practice (83%) and it should be the focus of research. So far, the focus has been on the Software Product Line case, which is much less common in practice.
- The Traditional Requirements Engineering approach to developing requirements is still most common (48% of responses), followed by the Use-Case-Driven approach (28%), whereas the Agile approach is least common (11%).
- The rate of adoption of requirements reuse varies among the teams that follow different approaches to developing requirements. Practicing reuse is much more common among the teams that follow the Traditional RE (61%) and Use-Case-Driven (56%) approaches as opposed to the teams that follow the Agile approach (33%).
- When reusing requirements for releases, the average reported reuse rate was 45%. However, 16% of the survey participants reported high reuse rates in the range of 80-100%, which indicates that implementing a highly effective requirements reuse process is achievable in practice.
- The teams that follow the Use-Case-Driven approach more commonly adopt reuse and reported much higher reuse rates (on average 64%) in comparison with the teams that follow the Traditional RE (43%) and Agile approaches (30%). Hence, when planning for adopting requirements reuse, developing product requirements based on use cases can be suggested as a preferred method.
- In contrast, the teams that follow the Agile approach reported the lowest reuse adoption and reuse effectiveness rates. This fact suggests an important opportunity to improve the Agile development methodology. In particular, the Agile process should address:
 - Separation of the stakeholder and product types of requirements. Currently, there is no agreement within the Agile community about which type of requirements the user stories represent.
 - Performing change impact analysis based on user stories.
- Separating the stakeholder and product types of requirements is not a common practice, however, it appears to be one of the factors that impacts effectiveness of requirements reuse, i.e., following the separation allows achieving higher reuse rates. Hence, when a team plans to adopt requirements reuse, it should start with analyzing the current practices and ensure that the concept of separating the stakeholder from product requirements is understood and agreed to.
- Most of the survey respondents informally identify requirements for reuse, where only 10% of respondents perform change impact analysis to do this, allowing them to achieve much better reuse effectiveness. When a project team plans to adopt requirements reuse or has already adopted it, they should include change impact analysis in the requirements process and ensure this practice is understood and followed for each software release.

About the Author

Yuri Chernak, Ph.D. is the president and principal consultant of Valley Forge Consulting, Inc. Yuri has worked for a number of major financial firms in New York leading IT Transformation initiatives and helping clients improve software requirements, software testing, and production management practices. Yuri has pioneered for financial applications on Wall Street a new discipline - aspect-oriented requirements engineering. He is a member of the IEEE Computer Society. He has been a

speaker at several international conferences in the US and Canada and has published papers in the IEEE publications and other professional journals. Yuri has a doctorate in computer science. Contact him by email: ychernak@yahoo.com

REFERENCES

1. Biggerstaff T and Perlis A (1989) Software Reusability. Addison-Wesley
2. Karlsson E (1995) Software Reuse: A Holistic Approach. John Wiley & Sons
3. Schaefer W, Prieto-Diaz R, Matsumoto M (1994) Software Reusability. Ellis Horwood
4. Jacobson I, Griss M, Jonsson P (1997) Software Reuse: Architecture, Process and Organization for Business Success. Addison-Wesley
5. Ezran M, Moricio M, Tully C (2002) Practical Software Reuse. Springer
6. Clements P, Northrop L (2002) Software Product Lines: Practices and Patterns. Addison-Wesley
7. Kang K, Sugumaran V, Park S (2009) Applied Software Product Line Engineering. CRC Press
8. Van der Linden F, Schmid K, Rommes E (2010) Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering. Springer
9. Pigoski T (1997) Practical Software Maintenance: Best Practices for Managing Your Software Investment. John Wiley & Sons
10. Grubb P, Takang A (2003) Software Maintenance: Concepts and Practice. World Scientific Publishing Co
11. Jarzabek S (2007) Effective Software Maintenance and Evolution: A Reuse-Based Approach. Auerbach Publications
12. Kang K, Lee J, Donohoe P (2002) Feature-Oriented Product Line Engineering, IEEE Software vol.19, no. 4, pp. 58-65
13. Moon M, Yeom K, Chae H S (2005) An Approach to Developing Domain Requirements as a Core Asset Based on Commonality and Variability Analysis in a Product Line, IEEE Transactions on Software Engineering, vol. 31, no. 7, pp. 551-569
14. IEEE Std.830-1998 IEEE Recommended Practice for Requirements Engineering
15. Kotonya G, Sommerville I (2003) Requirements Engineering, John Wiley & Sons
16. Armour F, Miller G (2001) Advanced Use Case Modeling, Addison-Wesley
17. Bittner K, Spence I (2003) Use Case Modeling, Addison-Wesley
18. Cohn M (2004) User Stories Applied. For Agile Software Development, Addison-Wesley
19. Leffingwell D, Widrig D (2011) Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise, Addison-Wesley
20. Chrissis M B, Kohrad M, Shrum S (2003) CMMI®: Guidelines for Process Integration and Product Improvement, Addison-Wesley
21. BABOK® Guide v.2. International Institute of Business Analysis (www.theiiba.org)
22. Wiegers K E (2003) Software Requirements, Microsoft Press
23. Aurum A, Wohlin C (2005), Engineering and Managing Software Requirements, Springer
24. Berenbach B, Paulish D J, Kazmeier J, Rudorfer A (2009) Software & Systems Requirements Engineering In Practice, McGraw Hill