A Model for Reusing Requirements Using Examples from Open Source Software

Jong Bae Kim*, Member, KIICE
Department of Computer, Soongsil University, Seoul 156-743, Korea

Abstract
The quality of requirements is one of the key factors in the success of a project. One of the studies on successful projects is the reuse of requirements. However, the rate of failed projects is about 70%, and these projects often fail because of improper requirements. The current techniques for software reuse may not perform requirement engineering appropriately or develop requirements having good characteristics. In order to improve this situation, we propose a model for reusing requirements. We expect that our model will contribute toward increasing project productivity using requirement reuse in existing projects, and reusing requirements that have good quality.

Index Terms: Requirements engineering, Reusing requirements, Domain engineering

I. INTRODUCTION

Requirement engineering is a sub-discipline of system and software engineering that is concerned with determining the goals, functions, and constraints of systems [1]. The importance of requirement engineering in ensuring the success of projects has already been shown by various studies. That is, the quality of requirements is one of the key factors in the success of a project. In this context, many methods for developing and managing good requirements have been studied [2-6], and one of them is the reuse of requirements [7-19].

Reuse technologies have already been applied in many industries and have become common practice. In software engineering, the limitations of the current paradigm of software reuse saw needs about reuse, and they were evaluated and synthesized to generate a new paradigm. Therefore, we pose some questions about current reuse techniques. First, "does the success rate of software projects increase with software reuse?" According to some research [20], the likelihood of the success of a given project is about 32 percent, and many projects are denoted as "challenged" or "failed". So, "why do so many projects fail?" According to some research [6], frequently (in about 70% of cases) the major factors in challenged or failed projects are uncorrected requirements such as omitted, incomplete, inconsistent, and changing requirements. That is, current techniques for software reuse might not allow for performing requirement engineering appropriately or developing requirements having good characteristics.

This study assumes that the current software crisis comes from not performing requirements engineering appropriately. Therefore, to solve this situation, we propose a model for reusing requirements. To achieve our goal, we set two objectives. First, we propose a concept for reusing requirements effectively and efficiently, which is called the
“domain element” concept. Second, we organize a procedure for reusing requirements. Our study does not cover specific techniques and tools for reuse.

We expect that our model could contribute toward increasing project productivity through requirement reuse in existing projects and reusing requirements that have good quality. It provides a foundation for increasing the success rate of projects.

The reminder of this paper is organized as follows. The related works are introduced at the next section. This includes analyzing existing works on reusing requirements and requirement engineering, and then we propose the concept of domain elements for reusing requirements in Section III. In Section IV, we explain the processes/activities for reuse by tailoring domain and requirement engineering. Section V represents case studies about our model. To validate the applicable feasibility of our model, we make a reuse repository by applying our model to 20 projects (project group A). To validate the effectiveness and efficiency of our model, we find requirements to be reused in other 20 projects (project group B) by using the repository. Then, to validate the improvement in reusability, we compare the requirements in project groups A and B. Finally, Section VI sums up our study, represents the limitations of our model based on a case study method, and proposes future work.

II. RELATED WORK

A. The Work of Buhne and Pohl [21]

Buhne and Pohl [21] propose domain requirement engineering, which is a process of developing and describing requirements for a domain. It is a process related to domain analysis in domain engineering. The term 'domain requirement' is a requirement for a specific domain, and to develop them, he proposes three steps: commonality analysis of requirements, variability analysis of requirements, and modeling common/variable requirements. This supports communication among the product manager, domain designer and application requirement engineer, and provides consistency regarding the variability of the product line. This study introduces an overview of domain analysis through steps, related techniques, modeling methods, and examples, and emphasizes commonality/variability analysis in reuse requirements. However, this might not provide a concept of reuse requirements.

B. The Work of Moon et al. [12]

Moon et al. [12] proposes an approach for developing domain requirements as the core asset based on commonality and variability analysis in a product line. This study suggests a meta-model of domain requirements as shown in Fig. 1, and proposes a process for developing domain requirement that consists of 4 steps. Step 1 is "scoping domain requirements". This has a goal of determining the boundaries of domain requirements and basic concepts and terms using specific domains. Step 2 is "identifying domain requirements". This has a goal of identifying domain requirements using a requirement-context matrix. Step 3 is "refining domain requirements". This has a goal of specifying the constraints between the domain requirements and writing the requirement specification. The final step is "constructing a domain-use case model". This has a goal to define requirement-use case matrix using the specified requirements. Moon et al. [12] contributes the proposal of a meta-model for commonality and variability of domain requirements and a process for developing domain requirements as core assets. However, the study of Moon et al. [12] might not provide specific techniques for identifying variation points and variants using a matrix of requirement-context and requirement-use case.

Fig. 1. Meta-model for domain requirements from Moon et al. [12].
C. The Work of Sutcliffe and Maiden [22]

Sutcliffe and Maiden [22] proposed a set of reusable abstraction concepts, modeling methods, and related tools for constructing a requirements reuse library, and this study suggested reusing domain requirements and knowledge by classifying domains and considering the relationship among them. This work proposes a meta-model of knowledge type for domain modeling, and proposes a framework for a reuse library and domain matching tool. However, the study of Sutcliffe and Maiden [22] might not separate between domain knowledge and domain requirements, and might not consider the relationship between domain and application requirements.

III. PROPOSING DOMAIN ELEMENTS FOR REUSING REQUIREMENTS

A. The Meta-Model of Domain Requirements

In domain engineering, domain analysis is an activity for eliciting reusable domain requirements in a domain [10]. The result of domain analysis, which is a set of domain requirements, is one of the input-artifacts for domain design and application engineering to adapt requirements for specific applications. The process for reusing requirements comprises the activity of making reusable domain requirements and the activity of adapting and translating them to application requirements. To elicit domain requirements, a meta-model of such requirements is proposed by Moon et al. [12] as shown in Fig. 1.

As in the meta-model, a domain requirement has some points of variation that are realized by variants. As Fig. 2, the requirements for specific applications will be made by adapting domain requirements to specific applications using variants (PRElement in Fig. 1).

B. Domain and Domain Requirement

The term "domain" of Trace's definition is a set of problems and solutions that cover similar applications [20]. Therefore, we consider the term "domain requirement" to be a connection between a problem and solution in a domain because the requirement is elicited as a middle artifact to develop solutions that can cover the problems. Fig. 3 shows the relationship among domain, problem, requirement, and solution.

C. Reorganizing Domain and Proposing Domain Elements

In this section, we consider one of most important issues, which is "how can domain scope be decided?" That is, we might decide whether a specific problem or requirement is included within the domain scope. To address this issue, we consider reorganizing problems and requirements to reuse them effectively and efficiently.
A Model for Reusing Requirements Using Examples from Open Source Software

A domain includes problems and solutions to be covered by similar applications. To reorganize the domain, problem, and solution, we will discuss two views of a domain: vertical and horizontal [17].

The vertical view is a business area that can classify domains such as military, medical, management, ordering, etc. The horizontal view is a functionality area such as a database, container library, workflow system, GUI, etc. Fig. 4 represents the vertical and horizontal views [18]. Table 1 represents the domain problems, requirements, and solutions considering the two views of the domain.

Table 1. The relationship among problems, requirements, and solutions considering vertical and horizontal views (example)

<table>
<thead>
<tr>
<th>Vertical view 1 (military)</th>
<th>Vertical view 2 (medical)</th>
<th>Vertical view n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal view 1 (DB)</td>
<td>Problem 1-1...n1</td>
<td>Problem 2-1...n4</td>
</tr>
<tr>
<td>Requirement 1-1...n2</td>
<td>Requirement 1-1...n2</td>
<td>Requirement 1-1...n5</td>
</tr>
<tr>
<td>Solution 1-1...n3</td>
<td>Solution 1-1...n3</td>
<td>Solution 2-1...n6</td>
</tr>
<tr>
<td>Horizontal view 2 (container library)</td>
<td>Problem 3-1...n7</td>
<td>Problem 4-1...n10</td>
</tr>
<tr>
<td>Requirement 3...n8</td>
<td>Requirement 3...n8</td>
<td>Requirement 4-1...n11</td>
</tr>
<tr>
<td>Solution 3...n9</td>
<td>Solution 3...n9</td>
<td>Solution 4-1...n12</td>
</tr>
<tr>
<td>Horizontal view n</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

IV. OUR PROPOSED PROCESS FOR DEVELOPING AND REUSING DOMAIN ELEMENTS

In this section, we consider the iterative processes of domain elements that comprise two areas. One area is a
process developing domain elements to ensure reusability, and another area is a process adapting and reusing domain elements to elicit domain/application requirements. We obtained the concept of the two areas from domain and application engineering in PLE (Product Line Engineering) and CD (Component Development) and CBSD (Component Based Software Development) in CBD (Component Based Development). According to this concept, the two areas are separated to find something reusable (such as core asset or component) and to reuse them. We propose our processes for the domain element as shown in Fig. 6.

Fig. 6. Proposing a meta-model of "domain element".

A. The Meta-Model of Domain Requirements

The goal of this process is to define domain elements that might include reusable requirements. That is, this is a pre-stage for reusing requirements, and these results are domain elements to support reusing requirements. This process is composed of 4 activities as follows.

1) Activity 1 - Identify Domain Problems, Requirements, and Solutions

The goal of this activity is to identify the basic information for developing the domain element. The basic information is a set of domain/application problems, requirements, and solutions as shown in Fig. 5. This information is elicited based on the problems and solutions of given applications. They are also elicited based on current problems in the real world.

Term Definition 2: Domain Information

Domain Information (DI): A set of domain problems, requirements, and solutions from given applications. This is used to develop domain elements as input artifacts.

· DI ⊂ Domain
· DI = (PDI, RDI, SDI), where
  PDI are problems solved by some applications;
  RDI are some requirements related to PDI;
  SDI are some solutions to meet RDI.

2) Activity 2 - Analyzing Relationships and Classifying Domains Considering Vertical and Horizontal Views

The goal of this activity is to prepare rationales for defining domain elements. The criteria for classification are vertical and horizontal views as shown in Fig. 4. One of the tasks in this activity is to establish a relationship among the elicited problems, requirements, and solutions in a domain. If the domain element of the same views already exists, we might add or refine domain information to the existing domain element.

3) Activity 3 - Defining Domain Elements

The goal of this activity is to define domain elements based on the results of the previous activity. After the domain problems, requirements, and solutions are classified, we are able to define domain elements. The commonality and variability of domain problems, requirements, and solutions are then analyzed to reuse them effectively and efficiently in applications. The domain element is composed of a specific problem (and related sub-problems), problem-related requirements, and solutions to meet these requirements, and the relationship among them. To reuse domain elements as basis unit in the reuse process, they should be defined independently from others. It is important that a domain element has independent characteristics because it could be reused for various domains. To reuse them, the commonality and variability of domain elements should be analyzed using several techniques [7-9, 22-24].

Term Definition 3: Domain Element

Domain Element (DE): This is an abstract concept consisting of a domain problem and related requirements and solutions to reuse them efficiently. DEs are related to each other. This is a subset of Domain Information.

· DE ⊂ DI, where the criteria of classification are vertical and horizontal views.
· DE = (PDE, RDE, SDE), where
  PDE is a problem(s) classified by vertical and horizontal views.
  RDEs are a set of requirements that have been prepared to reuse and are related to the PDE.
  SDEs are a set of solutions that have been prepared to reuse and could meet RDEs.
If there are already existing domain elements that contain similar domain views and information, they are added or modified using elicited information.

4) Activity 4 - Validating Domain Elements and Storing Them in a Repository
The goal of this activity is to ensure that domain elements are reusable and that reuse is manageable. To validate domain elements, we should ensure that they have good characteristics, such as cohesion and coupling. This has an effect on reusing them, and provides basis information to decide whether domain elements are correct or not. Second, a domain element is ensured to have correct information such as problems, requirements, solutions, and the relationship among them. Because the information for finding domain elements effectively and efficiently is based on the real world, a specific problem within the domain element might correspond to some requirements and solutions. Finally, the commonality and variability of each domain element should be reviewed to reuse and adapt them for different domains/applications.

After validating the domain elements, they are stored in a repository, which has a method for finding the correct reusable domain element efficiently. This repository and environment will be discussed in a future study.

B. Domain and Domain Requirement

The goal of this process is to reuse domain/application requirements with a domain element. In this process, the requirements are identified by performing domain analysis of a project, and are adapted to consider the project's characteristics [19].

1) Activity 1 - Determining a Project Domain and Scope
The goal of this activity is to select a project domain and determine its scope for finding reusable domain elements. The domain could be considered a specific area or abstraction of the real world having some problems to be solved by a project. The requirement could then be considered as the inter-relationship between the problems and solutions [2, 5]. Because problems are selected by a project, we assume that the project scope is determined by visualizing selected problems.

Term Definition 4: Project Scope

Project Scope (PS): This is an area that could include the problems to be solved by a specific project, related requirements, and solutions.

- \( PS \subseteq \text{Domain} \)
- \( PS = (PPS, RPS, SPS) \), where
  - PPS is a problem(s) interested in a specific project.
  - RPS is a set of requirements related to PPS.
  - SPS is a set of solutions to meet RPS.

2) Activity 2 - Selecting Domain Elements Related to the Scope
The goal of this activity is to select domain elements for identifying requirements in the selected scope. Some domain elements could be identified considering the relationships shown in the guidelines below. Because domain elements are a set of problems, requirements, and solutions that are classified as vertical and horizontal views of a domain and a set of related requirements, some candidate requirements are made by selecting domain element and applying the available variants of a project.

Guidelines for selecting domain elements as candidate elements
- A domain element that includes a PDE to be able to cover some PPS can be selected.
- A domain element that includes an RDE to be able to cover some RPS can be selected.
- A domain element that includes an SDE to be able to cover some SPS can be selected.
- A domain element that is considered to have similar vertical and horizontal views can be selected.

Term Definition 5: Candidate Requirements to be reused

Candidate Requirements (CR): The reusable requirements could be made by requirements of candidate elements and variants. A set of requirements to be made by applying variants is called Candidate Requirements.

- \( CR = (RPS, VP) \), where
  - VP is a set of variants applied by a specific project to RPS.

In identifying domain elements, the information for visualizing the domain/project scope is identified considering vertical and horizontal views, and this might provide some rationale for reusing requirements that have already been written.

3) Activity 3 - Adapting Domain Elements for Domain/Application
The goal of this activity is to adapt problems, requirements, and solutions in candidate elements considering the domain and project. To achieve the goal, this activity should be performed by completing some tasks to identify candidate variants that can be applied to varying points of domain elements and to select a variant considering the project/domain characteristic. As a result, the requirements for the domain and project are reused by applying the variant to the requirements of the domain element, which are then called reused requirements.
Term Definition 6: Reused Requirement by adapting domain elements

Reused Requirement (RR): The reused requirements are selected from Candidate Requirements.
- \( RR \subseteq CR \)

4) Activity 4 - Validating Domain/Application Problems, Requirements, and Solutions

The goal of this activity is to validate reused requirements from domain elements and to store them in a repository. Some methods and techniques can be used for validating the reusability and feasibility of the requirements. The result of validation might be stored with the used methods and techniques. It will provide some information for reusing the domain elements. These domain elements revised using the information will have good quality.

V. VALIDATION FOR OUR STUDY

Our study performs some case studies to validate the proposed domain elements to reuse requirements. The first case study is performed to validate our concept and developing domain elements. The candidate target is 20 applications in 4 categories in SourceForge.net, which is a portal of open source software (OSS), and we elicit some requirements of these projects. Using this requirement, domain elements are identified by applying our process for developing domain elements. We show the results of identified domain elements and of our concept.

The second case study is performed to validate the process for reusing domain elements. We select other applications that participate in a domain of a previous case study. We then create requirements using the previous results, which are domain elements of the first case study, and the process for reusing domain elements. We measure the reusability rate of the requirements that are developed using domain elements.

The third case study is performed to show evolutionary improvements in the reusability of requirements. Our processes are proposed to have more reusability than many applications used for domain elements. The third case study is similar to the second case study. However, the domain elements of the third one are improved using 20 other applications along with the second one. Therefore, we will compare the reusability rate between the second and third case study, and show improvements in the reusability of the domain element.

Table 2. Identifying domain element considering vertical/horizontal views

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>Business &amp; Enterprise</td>
<td>Communication</td>
<td>Development</td>
<td>Audio and Video</td>
</tr>
<tr>
<td>2</td>
<td>Customize</td>
<td>R047, R050</td>
<td>R077, R090, R098, R098, R104, R117, R121, R123</td>
<td>R180, R191</td>
<td>R224, R225, R251, R258</td>
</tr>
<tr>
<td>3</td>
<td>Globalization</td>
<td>R012, R059, R064</td>
<td>R071, R100, R127</td>
<td>R151, R159, R170, R218, R242, R245, R258</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Import &amp; Export</td>
<td>R004, R005, R008, R022, R028, R063</td>
<td>R099, R123, R129, R137</td>
<td>R142, R148, R150, R163, R203, R211, R229, R230, R247, R248, R249, R260, R261</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Integration</td>
<td>R014, R044, R048</td>
<td>R124, R198</td>
<td>R166, R168, R187, R188, R262</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Usability</td>
<td>R009, R019, R028, R043, R052, R053, R065, R073, R078, R119, R126, R191</td>
<td>R147, R152, R171, R194, R250, R252</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Accessibility</td>
<td>R003</td>
<td>R198</td>
<td>R232</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 7. Some requirements and features of specific software at SourceForge.net.
A. The First Case Study to Validate the Concept of Domain Elements and Their Development Process

To validate our domain element, we performed a feasibility study of our work. Considering our process of developing the domain element, we have selected 20 applications in SourceForge.net, and have elicited some requirements using the feature list in project site. Fig. 7 shows features of a software program named “KeePass”.

We have elicited 265 requirements of 20 applications for 4 categories in SourceForge.net. Our second step is analyzing the domain using horizontal and vertical views. We defined 4 vertical views like the 4 categories in SourceForge.net and identified 35 horizontal views by analyzing the relationship (such as semantic similarity) of words included in the requirements and naming them. Table 2 shows some requirements for each view.

The third step of our process is to define the domain element. One of the key activities is analyzing commonality and variability considering two views. Table 3 shows an example, which generates reusable requirements with Business & Enterprise (vertical view) and Customize (horizontal view) using R047 and R050 as follows:

R047: The user might make some custom fields.
R050: The system should support a personalization of the specific user.

R047: <The user> might make some custom <things> of <the system>.
R050: <The system> should support to make custom <things> to <the user>.

The “<>” in statement means a variation point to which different variants can be applied depending on the project characteristics. Such reusable requirements will be validated using appropriate techniques and will be refined to be reusable by applying the requirements of other projects. In this case study, 160 reusable requirements, 60 domain elements, 4 vertical views, and 22 horizontal views were defined by our process for developing domain elements.

B. The Second Case Study to Validate the Effectiveness and Efficiency of Domain Elements and the Process of Reusing Them

To validate reusing domain elements and their effectiveness and efficiency, we performed a second case study to develop application requirements using domain elements developed from the first case study. The specific applications were selected from SourceForge.net. We considered the project goal and elicited some requirements using domain elements. Fig. 8 is one of the examples that we selected from the Business & Enterprise category. We analyze the software considering views, so 13 domain elements and 32 domain-requirements are selected as shown in Table 3.

Table 3. The domain elements and requirements to be reused from the first case study in the business & enterprise category

<table>
<thead>
<tr>
<th>Horizontal view</th>
<th>Domain requirement</th>
</tr>
</thead>
</table>
| Customize       | <The user> might make some custom <things> of <the system>.
|                 | <The system> should support making custom <things> for <the user>.
| Globalization   | <The system> should support <multi-language>.
|                 | <The system> should be used <globally>.
| Import & export | <The system> should export to <File formats> files to <the purpose>.
|                 | <The system> should import to <File formats> files for <the purpose>.
|                 | <The system> should attach <some field and entry> to <the purpose>.
|                 | <The system> should make <some format> similar to <some file>.
| Integration     | <The system> should have <Architecture Style> architecture.
|                 | <The system> should have <Framework Type> framework.
|                 | <The system> should have <Engine Type> engine.
| Usability       | <The user> might use <User Interface Type> for <the purpose>.
|                 | <The system> should have <UI Quality> user interface for <the purpose>.
|                 | <The system> should configure <How>.
| Accessibility   | <The system> should support <SW Quality>.
|                 | <The system> should not install <Environment>.
|                 | <The user> can access the system <any~>.
| Manage & monitor| <The system> should manage <something>.
| Search          | <The system> should monitor <something>.
|                 | <The system> should audit <something>.
|                 | <The system> should search for <something>.
|                 | <The system> should sort <something>.
|                 | <The system> should tag <something>.
| Browsing        | <The system> should browse <something> where.
|                 | <The system> should support a self-browser.
|                 | <The system> should make <Browsing Type>.
| Look & feel     | <The system> should provide <GUI Elements>.
| Message         | <The system> should support remote control for <the purpose>.
|                 | <The system> should support <group type>.
| Navigation      | <The system> should support a way <how> to navigate to <target>.
|                 | <The user> can make some link from <link target> for <the purpose>.
| User account    | <The system> should support <user account type>.
Some variants considering project characteristics are applied to the domain requirements at the domain elements, and application requirements are elicited. That is, as project requirements are related to domain-elements, we consider whether to reuse domain requirements or not. Upon analyzing the result, 10 domain requirements of 32 domain requirements were found to be reusable and 33 application requirements of 45 application requirements can be elicited using domain elements.

To show the extent of reuse of the domain element, we use precision and accuracy metrics [25], which are the degrees of closeness to evaluate the effectiveness and efficiency of the subset elicited from the population. In our study, the metrics are calculated by analyzing the relationship between the elicited requirements and total requirements of either domains or applications. Table 4 represents the effectiveness and efficiency of the domain elements (elicited from the first case study) considering 20 applications.

It was found that the domain elements of the first case study have 20–40% precision. That is, using domain elements, at least 20% of the domain requirements would be reused. The domain elements have 33–80% accuracy, that is, the application-requirements elicited by using the domain elements are reused in the applications above 33% of the time.

C. The Third Case Study to Validate Improving Reusability of Domain Elements

The third case study was performed to validate the reusability of the domain elements. Our processes for developing and reusing domain elements include an activity...
for validating and storing the domain element to the repository to provide feedback and improve them. Therefore, reusability of domain elements might be increased as more requirements are collected. To validate them, we compare the domain elements of the first case study and second case study, and use the number of vertical and horizontal views and precision and accuracy as criteria. Table 5 represents the results of the third case study.

Table 5. The results of validating the reusability of domain elements

<table>
<thead>
<tr>
<th></th>
<th>Case study 1</th>
<th>Case study 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of applications</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>No. of categories</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>No. of requirements</td>
<td>265</td>
<td>504</td>
</tr>
<tr>
<td>No. of domain elements</td>
<td>60</td>
<td>68</td>
</tr>
<tr>
<td>No. of requirements within domain elements</td>
<td>160</td>
<td>284</td>
</tr>
<tr>
<td>Precision (low/mean/high)</td>
<td>20/30/40</td>
<td>31/44/52</td>
</tr>
<tr>
<td>Accuracy (low/mean/high)</td>
<td>33/55/80</td>
<td>45/61/76</td>
</tr>
</tbody>
</table>

From the comparison, it was found that domain elements that are generated from a large number of applications have higher values of precision and accuracy than those generated from fewer applications. Therefore, we validated the evolutionary improvement in reusability of our domain elements.

VI. CONCLUSIONS

Our study proposes the concept of domain elements for reusing requirements. The domain elements are developed by collecting information from various applications, analyzing them considering vertical and horizontal views, and defining domain elements using the results of the analysis. They are then reused in future applications by identifying the appropriate domain elements considering vertical and horizontal views of the future domain, applying variants of the (domain) requirement in the domain elements, and generating reused application requirements. In our study, we validate the concept of domain elements and related processes using 40 applications at Sourceforge.net.

There are some limitations of our study. First, our study does not propose techniques for analyzing commonality and variability of requirements in domain elements. Second, this study does not propose the concrete method for searching and applying variants to domain requirements. Finally, our study does not describe the environment for domain elements.

We will perform some future work as follows. First, to validate domain elements practically, we will perform many case studies and refine our results. Second, concrete methods and techniques related to our study should be developed. Finally, we will propose some tools and environments for automating our processes and managing domain elements.

REFERENCES


Jong-Bae Kim
received the M.S and Ph. D degrees in software engineering from SoongSil University, Seoul, Korea in 1999 and 2006, respectively. From 2001 to 2012, he was with E-Enterprise co.Ltd, Korea, where he was CEO. Since 2004, he has been with Seoul Women’s University and SoongSil University, and is presently an Adjunct Professor in the Department of Computer. His research interests are in Software Engineering, Open Source Software, and Information Security.