DREAM : Domain REquirement Asset Manager in Product Lines

Jihyeon Park, Mikyeong Moon, Keunhyuk Yeom
Department of Computer Engineering, Pusan National University
30 Changjeon Dong, Keumjeong Ku, Busan, 609-735, Korea
{witch03, mkmoon, yeom}@pusan.ac.kr

ABSTRACT

Product line engineering is a framework to support systematic reuse. The methodologies of product line engineering emphasize proactive reuse to construct high-quality, less costly products. Many software development technologies have been used in context of product line. The requirements for a product line are basis of software development as traditional system development engineering, and basis of deciding other core assets' property - commonalities and variabilities. Therefore, it is necessary to identify and explicitly denote the regions of commonality and points of variation at the requirement level. In this paper, we suggest a tool - DREAM(Domain REquirement Asset Manager) - to manage requirements that will be a core asset in the product line. Briefly, this tool supports the management of the commonalities and the variabilities of the domain requirement and stores the artifacts of the requirements engineering to reuse in the design process. Through this tool, the reuse of domain requirement can be enhanced.

KEYWORDS: Domain requirement, Requirements management, Product line

1. INTRODUCTION

The goal of product line engineering is to support the systematic development of a set of similar software systems by understanding and controlling their common and distinguishing characteristics [1]. That is, product line methodology focuses on systematically supporting reuse. The methodologies of domain engineering have been used to develop and manage core assets that can be reused. To date, much of product line engineering research has focused on the reuse of work products relating to the software's architecture, detail design, and code [2]. Requirements engineering is engineering to support all activities that related to requirements. Requirements are the foundation of all software planning and development activities. Requirements need to be automated by a tool because requirements become more complicated as developing systems in the context of product lines have recently increased. In the context of a product line, domain requirement is more complicated than general system requirements because they focus on and analyze the requirements of several systems in the domain. Domain requirement needs to be managed systematically using a generalized form so that they can be reused when reanalyzing the same domain or analyzing a subdomain as well as when developing new applications belonging to this domain. Thus, properties of the domain requirement need to be divided objectively. However, the currently used single system requirements management tools' ability to manage and generalize the domain requirement are limited.

In this paper, we suggest a tool - DREAM - to manage the requirements that will be a core asset in product lines. DREAM offers a standard for identifying commonalities and variabilities objectivity and enhances reusability by reducing complexity. Also, DREAM stores many artifacts of the requirements engineering. These artifacts will be reused in the design process of domain engineering. This paper is organized as follows: in the next section, related work is discussed, then in Section 3 the requirements management tool to support the domain requirement development process is, respectively, and lastly Section 4 concludes this paper

with a summary.

2. RELATED WORK

2.1. Background

Reusability of software is more efficient when software is planed and managed in a specific product line. Product line software development is visualized in two lifecycle models [3] as shown in Figure 1. The upper model represents the domain engineering process that has emerged to enable disciplines to analyze and synthesize domain information, which can be exploited for reuse [4]. Domain engineering that includes domain analysis, architecture development and reusable asset development produces models, architecture, and reusable components.

The lower model is an application engineering process that can analyze customer's requirements and develop new applications by assembling components already developed in domain engineering. New products are made in a product line by using artifacts of domain engineering and the requirements that can not meet by the assets of the product lines become feedback in domain engineering.

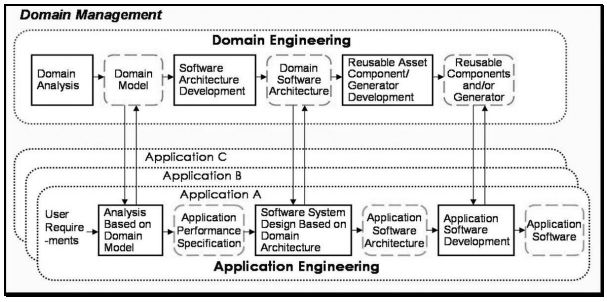


Figure 1 Development in product line

2.2. Requirement Engineering in Product Line Context

Requirements are the basis of all software planning and software development activities. The requirement engineering process is often described as cyclic with each cycle consisting of elicitation, analysis and validation activities. However, requirement engineering for software product families differs significantly from requirement engineering for single software products. Unlike single application requirements, the domain requirement is the common requirement of a system set in domain in the context of product line. Product line engineering develops and manages core assets, just like requirements, architecture, reusable components etc., which can be reused. To manage domain requirement as a core asset, the complexity of the domain must be reduced through identifying the commonalities and variabilities of the requirements and using them systematically in the domain engineering and application engineering processes.

2.3. Requirement Management Tool

As the software development process becomes complicated, more repetitions of and effort in the requirements elicitation and analysis phases are needed. Many automation tools about requirement management have developed. IBM's RequisitePro [5] and Telelogic's DOORS [6] are widely used at present.

IBM's RequisitePro supports the requirements engineering of RUP(Rational Unified Process) and supports database and MS Word to eliminate, manage and deliver requirements. RequisitePro supports a hierarchical structure between requirements and offers an automated trace function. Also, RequisitePro systematizes documents and data related to requirements while the system is being developed and controls and manages relationships among requirements, designs, usecases and test plans. RequisiteWeb manages requirements of web-based applications in a distributed environment.

Telelogic's DOORS offers traceability between applications and requirements to manage user's requirements systematically supplying basic form and database. Also, DOORS supports easy communication of the system's functions between analyst and user by mutually exchanging analysis, design tools and data.

DOORSnet is the internet based version and DOORSrequireIT is the MS word based version.

However, both RequisitePro and DOORS are limited in their ability to support requirements engineering in product lines because they manage requirements of single application.

3. TOOL SUPPORT IN THE DOMAIN REQUIREMENT DEVELOPMENT PROCESS

DREAM supports the domain requirement development process [7] shown in Figure 2.

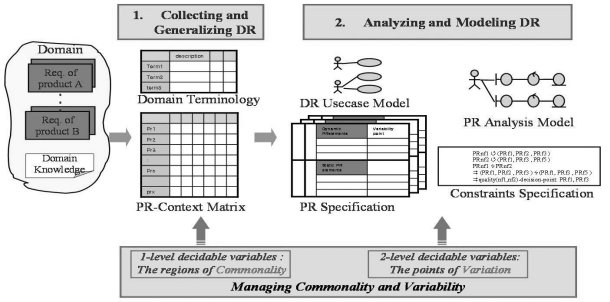


Figure 2 Domain requirement development process

3.1. Domain Requirement

Domain is defined by [8] and [9] as - "a family or set of systems including common functionality in a specified area." A domain requirement is defined as one common requirement that can be reused as a core asset of developing systems for a product line. That is, domain requirement describes the requirements of common skeletons in the systems, then describes somewhat abstracted types of slightly different parts in the systems.

In this paper, we suggest a meta-model for the DREAM. Requirement development based on the meta-model guarantees traceability between model elements [10]. Traceability between model elements helps the DREAM to maintain consistency among models. Figure 3 shows the meta-model of the DREAM

Domain requirement in the meta-model consists of six elements - domain, contexts, systems, PRs(Primitive Requirements), quality items and PRelements (marked by area in Figure 3). First, domain consists of a set of systems as a definition. Next, legacy systems composed of the same PRs can be generalized down to a single named context.

According to Wierzbicka, "any semantically complex word can be explicated by means of an exact paraphrase composed of simpler, more intelligible words than the original." [11]. These simpler and intelligible words are known as semantic primitives. The meanings of requirements can be decomposed into sets of semantic primitives. The divided semantic primitive is defined as the primitive requirement (PR) [12]. That is, a PR is used as a building block of a more complex refinement.

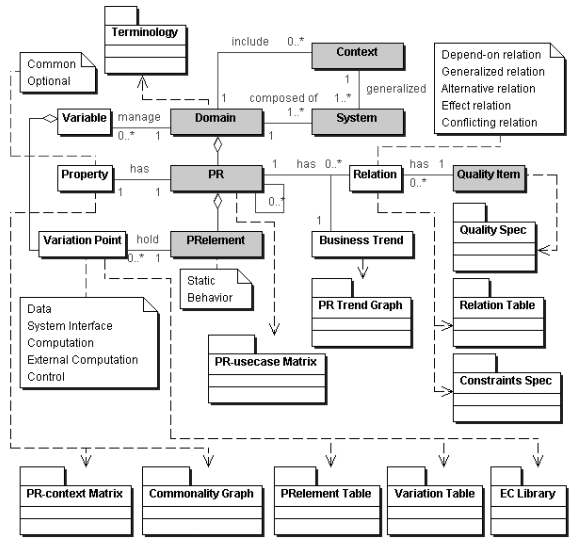


Figure 3 A meta-model of the DREAM

Quality items that are nonfunctional requirements identified in the quality spec. PRs have hierarchical structures on interior. PRs and quality items have a variety of relations – depend-on, generalized, alternative, affected, and conflictin. These relationships are identified by the relation table, and are described in the constraints spec.

One PR consists of several PRelements. PRelements are divided into behavior PRelements and static PRelements in the PRelement table.

PR's properties are characterized as either commonalities or optionalities according to frequency of their occurrence in the systems of a domain and at low levels, variabilities are identified. That is, first, commonalities and optionalities are identified and then variabilities are identified in the detail levels. Variability is classed again by data, system interface, computation, external computation and control, and is managed through the EC library and variation table. This limits the kinds of variabilities that can be identified at the requirements analysis level because at that level, its purpose is to specify the externally visible actions of the system. There is no need to define variabilities that appear at the detailed design level.

PRs are collected in the PR-context matrix. The common and optional properties of PR are determined according to the frequency that they appear in the systems in a domain. At this time, the commonality graph presents ratio of commonality objectively. While the domain analysis can be repeated several times, the properties of PR can be also change according to business changes. The "PR trend graph" presents the changes of the PR's property objectively. Domain experts have typically forecasted business trends, but the PR trend graph predicts the property of PR without the help of domain experts.

The PR-usecase matrix connects PRs and usecases, and supports traceability between requirements engineering and the design process.

3.2. Tool Support in Collecting and Generalizing Domain Requirement

To collect domain requirement, many legacy systems in a domain and the systems that will be developed must be analyzed. Even if a domain does not have a complete system, domain analysts find the domain that most closely resembles it and analyze systems. Domain requirement is not for a specific customer or a system, but should be predicted and analyzed in advance to satisfy potential customers and potential systems. Software engineering deals with existing experience as the most reliable data.

The priorities of the domain requirement are expressed by the commonalities, which are showed frequently to almost all systems and the optionalities, which are showed optionally.

A PR-context matrix [12] is displayed in Figure 4.

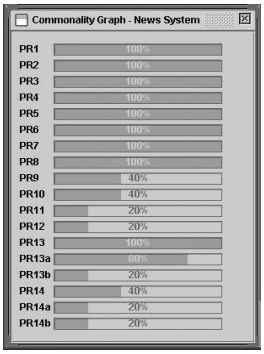
PR No.	PR	PROPERTY	RATIO(%)	Broadcast...	YTN	ET News	Chosun	Joins
PR1	Login	C	100	O	O	O	O	O
PR2	Logout	C	100	O	O	O	O	O
PR3	Register	C	100	O	O	O	O	O
PR4	Modify the member information	C	100	O	O	O	O	O
PR5	Withdraw	C	100	O	O	O	O	O
PR6	Set the grade	C	100	O	O	O	O	O
PR7	Write the news	C	100	O	O	O	O	O
PR8	Search the news	C	100	O	O	O	O	O
PR9	Write the opinion	P	40	O	X	X	X	O
PR10	Add the article at scrapbook	P	40	X	X	O	O	X
PR11	Search the scrapbook	P	20	X	X	X	O	X
PR12	Open the scrapbook to the public	P	20	X	X	X	O	X
PR13	Send the article	CV	100	O	O	O	O	O
PR13a	Send e-mail to the member		80	O	X	O	O	O
PR13b	Send mobile to the member		20	X	O	X	X	X
PR14	Sell news contents	PV	40	X	O	O	X	X
PR14a	Sell the newsletter		20	X	X	O	X	X
PR14b	Sell the article		20	X	O	X	X	X

Figure 4 PR-context matrix

All PRs of the legacy systems are merged into this matrix after checking for the existence of other legacy system's PRs. If a system has a PR it is marked 'O,' if not, it is marked 'X' where the column and row within the matrix meet. This matrix indicates statistically the properties of PR without depending on the experience and intuition of a domain expert.

DREAM that continuously updates the matrix to the database supports reuse when the same domain or subdomain is analyzed, and objectively predicts business trends by changing the PR ratio.

The most important role of the DREAM is to handle the properties of the commonalities and variabilities of the domain requirement. First, the PRs' commonalities and optionalities are identified in this step. Commonalities and optionalities are represented using frequency of which PR appears in the systems in a domain. Figure 5 shows the commonality graph that visualizes and suggests commonalities and optionalities of the domain requirement statistically.



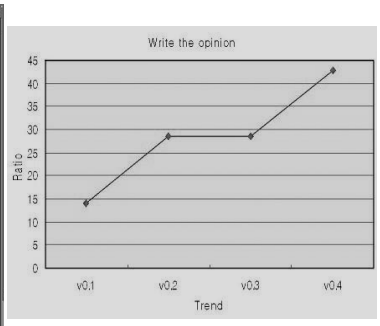


Figure 5 Commonality graph **Figure 6 PR trend graph**

The property of each PR can be changed through several reanalyses of a domain. To support this, DREAM offers a guide for predicting business trends by changing the PRs. Figure 6 shows that this PR's property is gradually changing from optionality to commonality.

3.3. Tool Support in Analyzing and Modeling Domain Requirement

The property of variabilities is managed in the domain requirement analysis step after commonalities and

optionalities are managed.

PR is composed by behavior PRelements and static PRelements, and variabilities are identified in each PRelement. There are five variation points - data, system interface, computation, external computation, and control variation point.

• Data – a particular data structure may vary from one system to another. This is mainly related to the variability of input or output data.

• System Interface – an external interface may vary from one system to another. This is influenced by data variability and the external operator.

• Computation – a particular function may exist in some systems and not in others. This is a variability which may occur in a process itself. For example, a process includes business rules or laws, or manipulates an external service.

• Control – a particular pattern of interaction may vary from one system to another. This may occur at a branch point of control flow.

Variabilities in computation and control can be identified in behavior PRelements, and variabilities in data and system interface can be identified in static PRelements.

Figure 7 shows an example of how variabilities about a member registration PR of a news system are handled. The Real Name Checking, which can be selected in the EC library is marked [vp] in EC category because it can be processed using an external service describing behavior PRelement. External computation's variabilities can be processed using an external service, which can be offered through a library because these are included in an application common domain and do not belong to a specific domain.

PR3	Register			
Ratio	100%			
Behavior PRelement	EC	computation	control	
1. Check the real name	vp			
2. Input basic member infomation				
...				
10. Pay the member registration fee			vp	
Static PRelement	data	interface		
1. basic member information	vp			
2. member register interface		vp		
3. payment system interface		vp		

Figure 7 PRelement table

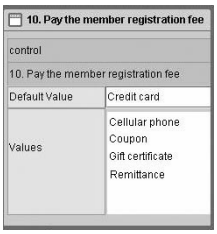


Figure 8 Variation table

Figure 7 shows that the member registration fee payment method can be handled with variability of control through [vp] item because various methods can be selected. When describing static PRelements, data and system interface variabilities are described with basic member information and member register interface regarding member registration. Figure 8 shows an example that handles variabilities of control that appear in member registration PR.

DREAM offers a quality spec to collect quality items that are nonfunctional requirements. Figure 9 shows the collected quality items and their priorities.

Quality No.	Quality Item	Priority
NFPR1	Perfomance	MEDIUM
NFPR2	Security	HIGH
NFPR3	Usability	HIGH

Figure 9 Quality spec

PRs and quality items have relationships. Figure10 shows the relationships that DREAM supports. The domain requirement's relationships are identified in the relation table and are described in the constraints spec (in Figure 11).

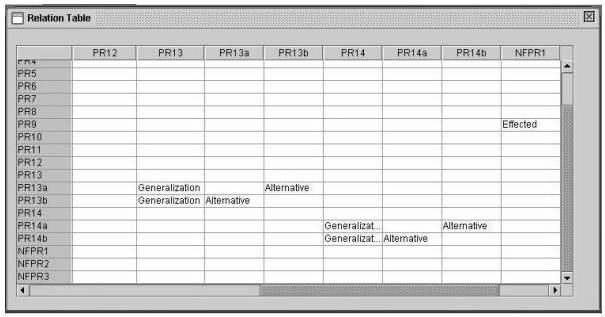


Figure 10 Relation table

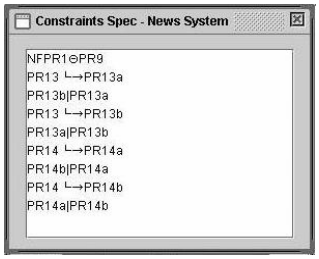


Figure 11 Constraints spec

3.4. Reusing Domain Requirement

DREAM stores many artifacts through the domain requirement development process. These artifacts will be reused in the design process of domain engineering as Figure

12.

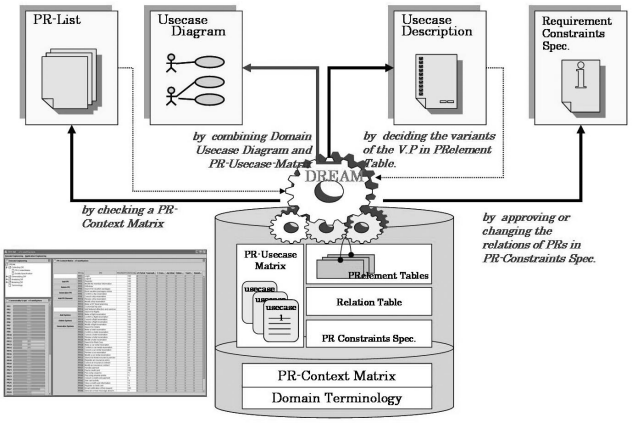


Figure 12 DREAM

4. CONCLUSIONS AND FUTURE WORK

This paper presented a requirements management tool - DREAM - that supports requirements engineering in product line engineering. That is, DREAM manages the properties of domain requirement as one of core assets. For this, various artifacts such as a PR-context matrix and PRelements table are used to analyze commonalities and variabilities of domain requirement objectively. Domain requirement is managed by the collection, generalization, analysis, and modeling processes. DREAM supports the analysis of commonalities and optionalities as domain requirement is collected, and supports the analysis of variabilities for the generalization, analysis and modeling processes. Also, reusablility is improved by keeping domain requirement that is a basis of the design process.

DREAM supporting product lines can be used to collect and analysis domain requirement, and also can offer business trends in the requirements objectively by feedback from the requirements of applications in a domain.

Our future research activities include supporting the relationship between domain requirements to manage changes and supporting traceability between requirements engineering and the other development processes after requirements engineering.

6. REFERENCES

[1] Muthig, D., Atkinson C., "Model-Driven Product Line Architecture," G. Chastek, editor, Software Product Lines: Proceeding`s of the Second Software Product Line Conference (SPLC2), San Diego, U.S.A., Aug. 2002, Heidelberg, Germany: Springer Lecture Notes in Computer Science Vol. 2379, 2002, pp.110-129.

[2] Faulk, S. R, “Product-line requirements specification (PRS): an approach and case study,” Proceedings. Fifth IEEE International Symposium on Requirements Engineering, 2001, pp.48 -55.

[3] Bristow, D., Bulat, B., and Burton, R., "Product-Line Process Development", Software Technology Conference, April 1995.

[4] Frank, S., "The Three "R's" of Mature System Development: Reuse, Reengineering, and Architecture", In The Fifth Systems Reengineering Technology Workshop 1995.

[5] IBM RequisitePro
URL:http://www.ibm.com/software/awdtools/reqpro/

[6] Telelogic DOORS
URL:http://www.telelogic.com/

[7] Moon, M. K., and Yeom, K., "An Approach to Develop Requirement as a Core Asset in Product Line", Bosch , J., and Krurger, C. (Eds.): ICSR 2004, LNCS 3107, July 2004, pp. 23-34.

[8] Creps, D., Klinger, C., Levine, L., and Allemang, D., ”Organization Domain Modeling(ODM) Guidebook Version 2.0”, Software Technology for Adaptable, Reliable System(STARS), 1996.

[9] Berard, M., Essays in Object-Oriented Software Engineering, Prentice Hall (1992).

[10] Streifferdt, D., “Traceability for System Families”, Software Engineering, In Proceedings of the 23rd International Conference on ICSE , May 2001, pp. 803 -804.

[11] Wierzbicka, A., Semantic Primitives, Frankfurt: Athenäum Verlag, 1972.

[12] Moon, M. K., and Yeom, K., “Domain Design Method to Support Effective Reuse in Component-Based Software Development”, 1st International Conference on Software Engineering Research and Applications, June 2003, pp. 149-154.