PAORE: Package Oriented Requirements Elicitation

Junzo Kato
Argo Soken Inc.
kato@foresee.or.jp

Motoshi Saeki
Tokyo Institute of Technology
saeki@se.cs.titech.ac.jp

Atsushi Ohnishi
Ritsumeikan University
ohnishi@selab.cs.ritsumei.ac.jp

Morio Nagata
Keio University
nagata@ae.keio.ac.jp

Haruhiko Kaiya
Shinshu University
kaiya@cs.shinshu-u.ac.jp

Seiichi Komiya
Shibaura Institute of Technology
skomiya@sic.shibaura-it.ac.jp

Shuichiro Yamamoto
NTT Data Corporation
yamamotos@rd.nttdata.co.jp

Hisayuki Horai
Celestar Lexico-Sciences, Inc.
horai@cl-scieneces.co.jp

Kenji Watahiki
Tokyo Institute of Technology
watahiki@se.cs.titech.ac.jp

Abstract

We propose a new requirements elicitation method in such a domain of ERP, CRM, and SCM by using specifications of several existing package software. We have analyzed the requirements elicitation processes of experienced analysts in a specific domain, and found that they clarify requirements by referring the specifications of existing packages that seem to be satisfied with customer's needs. This process can be formulated into two sub-processes: 1) package selection, where an analyst compares the customer's needs with functions/non-functions of packages and selects the suitable candidates of packages; and 2) requirements evolution, where he examines the selected packages with his customer and an approved part of specifications of packages are added into their requirements. The proposed method, called PAORE (PAckage Oriented Requirements Elicitation method) is designed based on the analysis. We applied this method to a simple but realistic example of Web-based Sales Supporting System and assessed it.

Keywords: Requirements Elicitation, Software Packages, Domain Knowledge

1 Introduction

Precise requirements analysis needs both domain knowledge and techniques of requirements elicitation. Because of the diversity of software to be developed and the extension of customer's needs, we have to research and establish a method that even a software engineer without domain knowledge and/or software development experiences can succeed in a precise requirements analysis.

We have proposed a method that supports an analyst without requirements analysis techniques to interview stakeholders[4]. Although many researches of domain engineering intend to help an analyst without domain knowledge[1], there does not exist any methods to construct a practical domain knowledge base. In this paper, we regard existing software packages as a source of practical domain knowledge because such software packages reflect functional and non-functional features of the domain, and we found that experienced software engineers precisely elicit software requirements with them. We propose a method named PAORE (Package Oriented Requirements Elicitation process) for eliciting precisely software requirements with existing software packages.

In PAORE, domain knowledge is elicited through investigating specifications of software packages in advance. A requirements analyst selects suitable software packages corresponding to customer's requirements, then elicits and specify requirements in detail by both showing the concrete specifications of the selected packages to his customer, and adding the customer's specific requirements which does not appears in the specifications.

Such a domain knowledge base of practical, systematic, and high quality can be easily constructed by collecting domain knowledge from existing software packages.

Figure 1 shows software packages as domain knowledge in PAORE. In Figure 1, package A and B have similar features in a domain. Partial domain knowledge, i.e. a table of relationships between packages and a list of functional features is derived from the specification of each package, and then they are systematized as total domain knowledge. In requirements analysis, an analyst can add customer's specific requirements and precisely specify initial require-

Proceedings of the Tenth Asia-Pacific Software Engineering Conference (APSEC'03)
1530-1362/03 $ 17.00 © 2003 IEEE

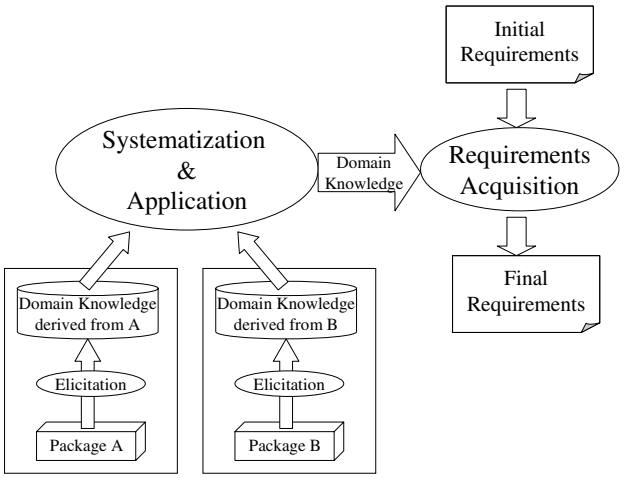

Figure 1. Software Packages as Domain Knowledge in PAORE

Table 1. PSM Level 0: "Hidemaru" and "Ichitaro" are famous software products in Japan. The former is text editor, and the latter is word processor.

Package \ Feature	text editor	word processor	DTP	Graphic Editor
Hidemaru	X			
Emacs	X			
Ichitaro		X		
LaTeX			X	
Tgif				X
Power Point (MS Office)			X	X
Word (MS Office)		X	X	
Visio				X

ments.

The rest of this paper is organized as follows. In section 2, we show how to represent knowledge in PAORE. In section 3, we explain how to elicit requirements by using PAORE. We show an example of a requirements elicitation by PAORE for demonstrating its usefulness in section 4. Finally, we summarize our current results by referring related works, and show our future works.

2 Domain Knowledge in PAORE

As mentioned in the first section, domain knowledge in PAORE is represented using information acquired from multiple packages. In this section, we show how to represent such knowledge in PAORE. Most software products consist of several software packages that can work alone. Packages in the same product sometimes complement or overlap their functions and/or qualities with each other. For example, a package MS-Office consists of a spread sheet, a word processor, a presentation software, and so on. Any package in MS-Office can be used for writing simple diagrams. In PAORE, we regard a package as a set of basic units of domain knowledge. In order to derive and systematize domain knowledge using multiple packages, domain knowledge in the packages should be easily compared and combined.

In PAORE, we introduce PSM (Package Solution Mapping) for this purpose. Table 1 and 2 are examples of PSM. Each line in PSM corresponds to a package, and each column corresponds to a feature. A feature is used to unify the different terms in the different packages that denote the same concept. Normally, a feature represents a function or a quality attribute of software, and it is named by a term commonly used in several packages, e.g. "font section", "output format", and so on. Features are also used to characterize a domain. For example in Table 1, relatively abstract features are used for characterizing a domain "Technical Documentation". In this example, we regard "text editor" as software with which we can only edit plain text files. We regard "word processor" as software with which we can edit complex document files embedding figures and/or tables.

Features in Table 2 are more concrete than those in Table 1. Features "word processor" and "DTP" in Table 1 are decomposed into nine features in Table 2. Note that not all packages in Table 1 appear in Table 2 because some packages are not related to these two features in Table 1. We may decompose features in Table 2 furthermore for representing these features in detail.

Each value in the cells of PSM shows the relation between a package and a feature. A mark "X" means that the package has the feature, while a blank cell means that the package does not have the feature. The advantages for representing domain knowledge by PSM are as follows.

- We can avoid difficulty for reusing domain knowledge caused by the vast inessential variety in technical terms because we introduce features in order to identify concepts among different sources of knowledge.
- We can decrease dogmatic dependence on a specific package, because multiple sources of knowledge can be used impartially at the same time. Note that we assume that such dependence is inconvenient for accurate and creative requirements elicitation.
- We can easily find packages that support the same or similar feature because of the matrix form of PSM.
- We can easily maintain hierarchy of domain knowledge, because features are classified into different lev-

Table 2. PSM Level 1: decomposing "word processor" and "DTP" features.

Package \ Feature	input/edit	font selection	mathematical equations	spelling check	grammer check	multicolumn	output PS and/or PDF format	edit figure/graphs	insert figure/graphs
Hidemaru	X								
Emacs	X			X					
Ichitaro	X	X	X			X	X	X	X
LaTeX		X	X			X	X	X	X
PowerPoint (MS Office)		X	X				X	X	X
MS Word (MS Office)	X	X	X	X	X	X	X	X	X

els of abstraction.

In the practical software development, an analyst improves his skill of requirements elicitation and analysis through experiences of using a specific software package. In PAORE, we do not only use this fact, but also extend it as follows.

- Experiences using different packages can be combined.
- Such experiences can be easily reused with an explicit representation in PSM.

In this paper, we do not mention how to construct PSM from specifications of actual packages.

3 PAORE Process

3.1 Requirements elicitation with PAORE

Figure 2 shows both requirements elicitation and requirements specification with PAORE. At first, a requirements analyst makes an initial list of requirements items, called "requirements item list" or RL, using some requirements elicitation method, such as interview. The initial requirements item list is an input of PAORE process. The output of PAORE process is an evolved requirements item list. Finally, the analyst makes a Software Requirements Specification (SRS) with this evolved list. In this paper, we focus on the PAORE process hatched in Figure 2.

3.2 Overview of the PAORE Process

The PAORE process is a method for evolving a requirements item list based on PSM. We assume that PSM has been made from the software packages before an analyst starts the PAORE process. He evolves the initial RL to the final RL by adding some features in the PAORE process. During each evolution, he compares the features in RL with those in the prepared RSM. An analyst makes Feature Requirements Mapping (FRM) for comparing these features. Each row of FRM corresponds to a feature in PSM, while each column of FRM corresponds to a requirements item

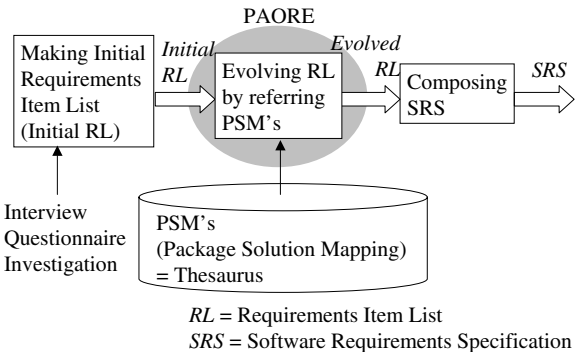

Figure 2. PAORE in Requirements Analysis

in RL. For example, the requirements item "IEEE format of research paper" in Table 3 is related to two features in PSM "font selection" and "multicolumn". Each package in PSM is linked to the actual instance of the package software. When an analyst adds a feature of a package to RL, he refers to the instance of the package by using the link. It is assumed that PSM does not change in the PAORE process.

Figure 3 shows the evolution of RL in the PAORE process. The PAORE process comprises the iteration of evolution cycle. There is a three times of iteration in Figure 3. As the iteration proceeds, RL is gradually evolved. In each cycle of the iteration, items in RL are refined and new requirements items are added to RL. Furthermore, in each cycle, the relevant features are extracted from PSM and added to FRM. These steps of evolution continue until all relevant features are derived from PSM. The final RL consists of detailed requirements items that are related to features in PSM. Features are defined hierarchically in PSM. According to the structure, we can construct a hierarchical structure of requirements items in RL.

A cycle of evolution consists of the following tasks:

1. Selecting the packages
   Select features which is relevant to RL, find PSM Level i whose rows include the features, and packages which have the feature are found in PSM Level i.

Table 3. An Example of Initial RL

Purpose	Easy to make English conference papers
Goal	platform independent, spell and grammar checking

FR1	IEEE format of research paper
FR2	write and edit English text
FR3	Printing with PS and PDF format
FR4	Editting figures and graphs
FR5	Editting mathematical equations
FR6	Spell checking
FR7	Grammar checking
NFR1	Availability
NFR2	not so expensive
NFR3	Delivery in time

2. Evolving RL
   Make FRM from RL and PSM Level $i+1$, create new requirements items related to the features in FRM, and add the new requirements items to RL. In other words, requirements items related to features in PSM Level i are refined to requirements items related to features in PSM Level $i+1$ in this step.

3.3 PAORE process

In this section, we illustrate the PAORE method with a simple example of writing conference papers. In 3.3.1, we describe how to select software packages. In 3.3.2, we describe how to evolve requirements item list with FRM.

3.3.1 Software Package Selection

We assume that an initial requirements item list is given by using a certain requirements eliciting method. Using this list, we will select packages as the following steps:

(1-1) We decide keywords to satisfy the initial RL. Nouns and noun phrases in the initial RL is helpful for the decision.

(1-2) We select features in PSM that are same as or similar to the keywords.

(1-3) We select packages which have the features.

Assume an initial RL shown in Table 3. We decide the following keywords to satisfy requirements items in Table 3.

- Word processor, text editor with DTP tool
- Graphic editor
- Editor for mathematical equations
- DTP

Keywords should be selected as words that precisely describe package features. Then, features same as or similar to the keywords will be selected. In this example, word processor, text editor, DTP, and graphic editor are selected. Next, packages that provide several numbers of these features are selected. PSM Level 0 is shown in Table 1. Since "editor for mathematical equations" are regarded as a detailed feature of word processor or DTP, this feature is not appeared in this table.

3.3.2 Evolving RL

We evolve RL based on the following guideline by deriving features in lower PSM from features of selected packages.

- We add new requirements that the customer wants if the new requirements are consistent with the existing requirements in RL.
- Since features are defined hierarchically in PSM, features in RL can be detailed into more precise sub-features and the sub-features will be added to RL.

The steps of evolving RL are as follows:

(2-1) We identify lower PSM for selected packages and their features. Table 2 shows the lower PSM that is PSM of level 1, for selected packages and features of word processor and DTP.

(2-2) Then, we make FRM. FRM made from Table 3 and Table 2 is shown in Table 4. In table 4, FR4 in the initial RL is not appeared, because FR4 is not related to features of "word processor and DTP" but to features of graphic editor.

(2-3) We add new requirements items to the RL with the FRM. New features not in the RL but in the FRM will be added to the RL, if the customer confirms them. An example of added features is shown in Table 5. FR8 and FR9 in table 5 are new functional requirements which are not in the initial RL. NFR4, 5 and 6 are new non-functional requirements that are not in the initial RL.

We repeat the above steps of evolving RL and finally obtain a detailed RL.

4 Experimental Analysis

In this section, we analyze a process where an analyst actually elicited requirements from a customer based on the method mentioned in Section 3. This example of a requirements elicitation process is for development of a sales support system for personal computer systems (PCs) and it is for practical use. The aim of this case study is to assess the

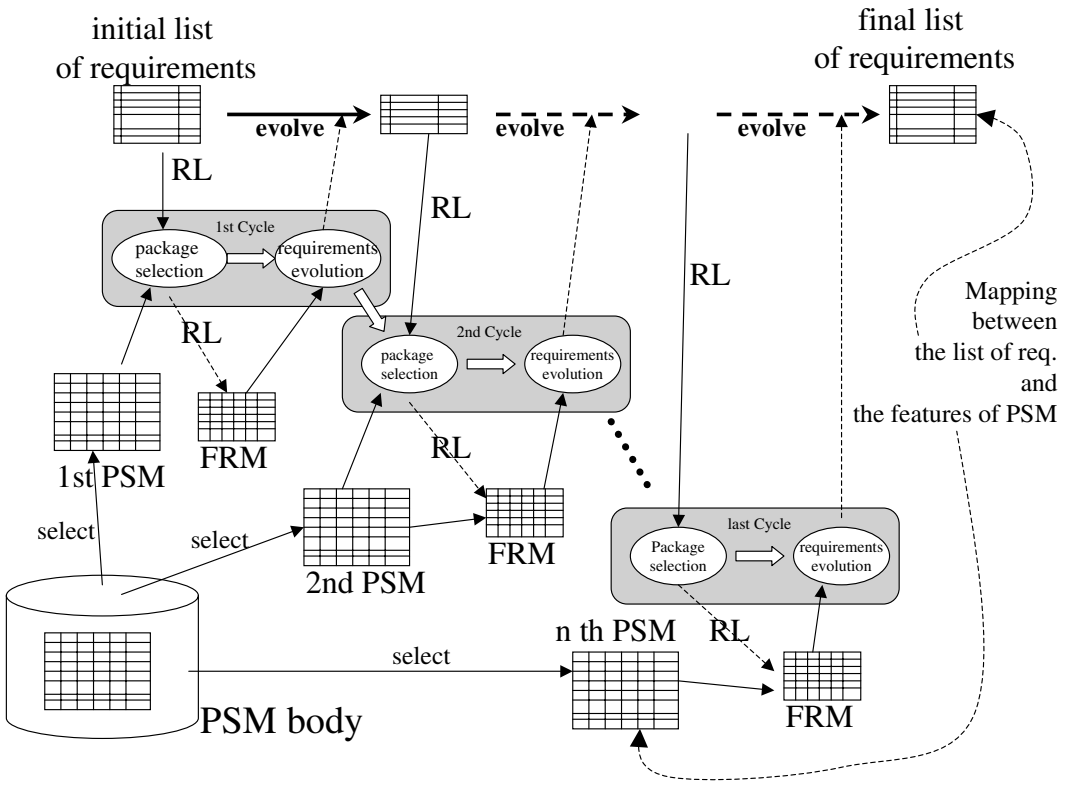

Figure 3. PAORE Process

Table 4. FRM for word processors and DTPs

	FR1	FR2	FR3	FR4	FR5	FR6	FR7	FR8	FR9
input/edit		X							
font selection	X								
mathematical equations					X				
spell check						X			
multicolumn	X						X		
output format			X						
edit figures/graphs								X	
insert figures/graphs									X

Table 6. Initial Requirements Item List

	Functional requirements
F1	A buyer can see a list of goods on sale by using his PC.
F2	A sales staff can make system specification of PCs.
F3	A system can dispay a list of goods and their prices by user's retrievals.
F4	A user can retrieve goods and display the results of retrievals.
F5	A system can make estimates and store them.
F6	A system can revise and modify estimates.
F7	A user can order and inquire PCs by E-mail.
F8	A user can update and check a list of customers.
F9	A system can update and maintain a list of goods on sale.

effectiveness of PAORE method by examine how many degrees of a requirements item list is evolved from an initial one.

4.1 Experiment

We assigned the role of an analyst, i.e. a subject of our experiment, to a software engineer who has worked for a system integration company. A planning manager of a sales company for PCs listed up a rough sketch of requirements of the system to be developed. This list is shown in Table 6 and we considered it as an initial requirements item list.

We provided the three levels of PSMs shown in Table 7 (a), (b) and (c) respectively. An expert software engineer developed these PSMs. They are the domain knowledge that the analyst could use in this experiment. The PSMs shown in Table 7 (b) and (c) are suitable only for the estimate function of the developed system, because we just focus on requirements elicitation for the estimate function in this paper on account of space. For example, the feature "maintenance contract" denotes that the maintenance fee is calculated and included in the estimation. As shown in Ta-

Table 5. Evolved RL

FR1	IEEE format	Multicolumn format	Two column format
		Selecting Fonts	Selecting from 9pt to 18pt
FR2	English	input and edit	insert, update, delete
FR3	Output in PS and/or PDF	Selecting an output file format	PS and/or PDF format
FR4	Figures and Schemas	General Figures, Schemas, templates	
	UML	Input and edit UML Diagrams	
FR5	Mathematical Expression	Input and edit methematical exp.	
FR6	Spel Checking	Spell Checking of English words.	
FR7	Syntax Checking	English Syntax Checking.	
FR8		Insert Figures and Tables.	
FR9		Input and edit Figures and Tables.	
NFR1	Availability	Use anywhere.	
NFR2	Cheap	Low Price	Less than 50000 JPY
NFR3	Prompt Delivery	Delivery Time (Period from order to delivery)	Two weeks.
NFR4	Stable	Environments and Stability	
NFR5	Simple operations	Input, output, edit.	
NFR6	OS	Unix, Windows	Windows2000

ble 7, the feature "Estimation" in PSM level 0 is refined into PSM level 1 (Table 7 (b)) and the feature "Estimated items" is also refined into PSM level 2 (Table 7 (c)).

The analyst applied PAORE method and made the requirements item list complete. In particular, we focused on the process where he elicited requirements items that were not included in the initial requirements item list, and added them to the requirements item list.

4.2 Experimental Results

Table 8 shows the FRMs of the estimation feature. The FRMs was constructed according to the hierarchical structure of the PSMs (shown in Table 7) during the experiment. The analyst's activities concerning an "estimation" were as follows:

1. Developing FRM Level 0 from an initial requirements list and PSM Level 0. The FRM Level 0 is shown in Table 8(a). An occurrence of the mark "X" stands for the existence of relationship between a feature in row and a requirements item in column. For example, requirements item F2, i.e. "A sales staff can make system specifications of PCs", is related to the feature "estimation". This relationship is based on the fact that, for another example, the customer often decides the PC specification with its price. Four requirements items F2, F3, F5 and F6 in the initial requirements item list are related to the feature "estimation".

2. Developing FRM Level 1 from PSM Level 1 and the four items appearing in FRM Level 0. FRM Level 1 is shown in Table 8 (b). In this step, the analyst found the features that are not in the initial requirements item list such as "Sending temporary estimates", "Sending order sheets" and "Estimated items". Gray-colored entries in the table stand for these features and they are the candidates that are newly added to the requirements item list.

3. Developing FRM Level 2 from PSM Level 2 with features including three new features found in the step 2. In Table 8 (c), the analyst picked up the feature "Estimated item" and constructed FRM Level 2 for the feature by using PSM Level 2 shown in Table 7 (c). The analyst added the feature "Ordinary items" to FRM Level 2, because, in a general commercial transaction, estimation items are a set of ordinary items such as the date of issue of an estimate, the term of validity, the publisher of an estimate, an estimated item, a unit price, quantity, the amount of money, the discount amount of money, and so on. The other three features, i.e. "maintenance contract", "installation", and "delivery schedule and calculation of actual delivery date", were also added to FRM Level 2. The analyst detailed the four features in of Table 8 (c) to four new requirements items. Finally, he added them to the requirement item list.

The above steps are iterated and the RL is gradually being evolved. The analyst went back to the other features in level 1 such as "Sending temporary estimates" and "Publishing order sheets" appearing Table 8 (b) and refined them by using their PSMs. He obtained the detailed features "Sending

Table 7. PSMs

(a) PSM Level 0

Feature / Package	Retrieval	Estimation	Receiving orders	Customer Management	Sales Management
A	X	X	X	X	X
B	X	X	X	X	X
C	X	X			
D	X	X			

(b) PSM Level 1: for Estimation

	Specifying configurations of goods	Simulating configurations of goods.	Creating estimates	Storing estimates	Retrieving estimates	Recording changes of estimates	Sending estimates	Sending temporary estimates	Publishing order sheets	Estimated items
A	X	X	X	X	X	X	X	X		X
B	X	X	X	X	X	X			X	X

(c) PSM Level 2: for Estimated items

	maintainance contract	installation	delivery schedule and calculation of delivery date	ordinary items (goods and prices)
A	X	X	X	X
B	X	X	X	X

temporary estimates by E-mail", "Publishing order sheets with digital signature", and so on.

Consequently, eleven requirements items are listed in the final requirements item list as shown in figure 4. Four of them are in the initial requirements item list, while the other seven requirements items are not. These seven requirements items represent customer's requirements that he did not find by himself or forgot to mention when the initial requirements item list was made. Four of the seven are the results of step 3, while the other three are the results of step 2. The followings are the final requirements list that the analyst obtained.

4.3 Findings

The results mentioned in section 4.2 show that the requirements items in the final requirements item list are about twice as many as in the initial requirements item list. We can conclude that PAORE has an effect on eliciting the requirements that a customer hardly finds by himself or he forgets to mention, and an effect on refining abstract requirements into more concrete ones. The requirements items labeled with "(new requirements item)" did not appear in the initial list of Table 6, i.e. newly elicited and added, and the other items were the results of refining the initial items. As a result, the final requirements items are a collection of the suitable features that different packages have. For example in Table 7 (b), package A has the feature "Sending temporary estimates", while package B has the feature "Publishing order sheets". It means that PAORE can be considered as a method for selecting and collecting the "best" features for a customer from various packages.

After the experiment, we interviewed the analyst. The finding from the interview are summarized as follows:

1. Requirements elicitation based on PAORE allows the analyst to provide the concrete product image in early stage for his customer. It was useful for the analyst to communicate with his customer.
2. The analyst had a package selection task once and performed requirements evolution with only one cycle in this case study. He did not take the further refinement cycle because the packages A and B were suitable for the requirements of his customer in almost all portions, and he could write the requirements specification concretely.
3. By using PSMs, the analyst could find several unrealizable requirements items in the initial requirements item list, and could delete them.

5 Discussion

PAORE is based on the functional approach using the list of functional requirements as well as non-functional re-

Table 8. Produced FRMs

(a) FRM Level 0

	F1	F2	F3	F4	F5	F6	F7	F8	F9
Retrieval	X			X					
Estimation		X	X		X	X			
Receiving Orders							X		
Customer Management								X	
Sales Management									X

(b) FRM Level 1: for Estimation

	F1	F2	F3	F4	F5	F6	F7	F8	F9
Specifying configurations of goods		X							
Simulating configurations of goods			X						
Creating estimates				X					
Storing estimates				X					
Retrievaling estimates					X				
Recording changes of estimates					X				
Sending temporary estimates									
Publishing order sheets									
Estimated items									

(c) FRM Level 2: for Estimated Items

	F1	F2	F3	F4	F5	F6	F7	F8	F9
maintainance contract									
installation									
delivery schedule and calculation of actual delivery date									
ordinary items (goods and prices)									

quirements. The reason why we use this approach is that function oriented requirements elicitation is more efficient than other approaches[3]. The results of the experiment show that the PSM developed by experienced analysts has the following properties.

- Packages have features to develop the desired requirements item list.
- The specific domain characteristics are able to describe using the common terms.
- PSM has features which have unique meaning.

The PSM also helps an analyst understand the domain specific characteristics selected from the overall functional requirements. The conditions to apply PAORE are as follows:

- It is difficult to require an analyst to have high expertise on the domain.
- There are multiple popular packages on the problem domain, and each of them has comprehensive document such as user's manual.
- Analysts have some experience of requirements analysis.
- There is a domain term dictionary.

If these conditions are satisfied, PAORE helps an analyst elicit requirements even if he does not have any knowledge and application experience on packages. As PAORE provides a means to select the best of breed features from packages, it has the following merits and demerits.

In case of PAORE, PSM describes the important portions of domain knowledge, although it does not deal with the detail of each package implementation.

To understand package specification terms completely, it may need to execute packages. However these implementation dependent requirements are restricted to small portions of the total requirements. In addition, these kinds of term understanding problems can be solved in the course of developing requirements specifications.

Merits:

- Requirements can be considered concretely in the scope of PSM features.

1. Sales staffs can specify customer's configuration of PC (initial requirements item F2).
2. Customers (buyers of PCs) can simulate their configuration of PCs (initial requirements item F3).
3. Sales staffs can create and maintain estimates for customers (initial requirements item F5).
4. Sales staffs can retrieve estimates and records their changes (initial requirements item F6).
5. An estimate has ordinary items. (new requirements item concerning initial requirements item F5).
6. A maintenance contract can be add at an estimated item (new requirements item concerning initial requirements item F5).
7. Customers can choose an installation matter of software as an estimated item (new requirements item concerning initial requirements item F5).
8. Customers can add the schedule of a date of delivery to a set of estimated items and the actual delivery data is automatically calculated. (new requirements item concerning initial requirements item F5).
9. Sales staffs can send the estimate by the E-mail with digital signature (new requirements item).
10. Sales staffs can publish the order sheet with digital signature (new requirements item).
11. Sales staffs can send a temporary estimate by the E-mail or fax (new requirements item).

Figure 4. Final Requirements Item List

- The best of breed requirements item list can be developed as it covers minimum requirements on the domain.
- PSM is useful for the communication with customers because analysts consider requirements using PSM based on domain terms but package implementation details.

Demerits:

- Consistency of the requirements item list should be checked. As RL is extracted from different packages, inconsistency among requirements items in it may occur.
- Although each package assures the feasibility of their requirements, it is necessary to confirm the total feasibility of RL selected from different packages.
- It is difficult to elicit requirements that are not out of the range in the existing packages such as new business models and non-functional requirements.

Although details are not mentioned here, information other than package specifications can be used in case of non-functional requirements elicitation and realization of requirements. The analyst of our experiment interviewed his customer about performance and safeness based on non-functional quality rules of ISO/IEC9126 standards, because concrete non-functional requirements were not appeared in the course of the experiment. For example, the necessity of the authentication by electronic signature was asked. This example shows that auxiliary knowledge source will be useful for requirements elicitation.

6 Related Work

The requirements elicitation process we proposed is based on reuse of software packages. Some methods to select a software package or a commercial componentware called COTS from requirements specification have been proposed. All of them are methods to select a suitable package or component based upon the given requirements elicited completely in advance. In [9], they proposed a complete lifecycle Meta model of COTS software acquisition (SAMM). SAMM includes a selection phase and an implementation phase. In [10], they proposed a concurrent iterative method of requirements acquisition and the COTS selection (PORE). In the course of PORE process, various techniques and activities are provided for requirements elicitation and COTS selection in the context-driven manner. Its process is flexible, but complex as they reported. Furthermore, Maiden et al. reported some lessons learned concerning requirements elicitation and COTS selection from projects that PORE was applied[8]. OTSO by Kontio et al. includes a systematic method to evaluate and select packages from viewpoints of costs and effectiveness[6]. For other instances, CAPS by Ochs et al.[11] and the research results by Franch et al.[2] include a metrics in order to evaluate and select components. Another research includes a proposal of requirements elicitation, documentation and evaluation methods for selection in the middleware domain[7]. All of these methods and techniques require

complete requirements of a customer and the feasibility of the requirements for evaluation and selection of packages and components. In PAORE, requirements are elicited gradually in course of package evaluation and selection. PAORE is a suitable method in a real development where the complete user requirements cannot be elicited beforehand.

Requirements elicitation methods based on domain knowledge, such as Requirements Apprentice[13], have been proposed. One of the most difficult issues in a domain-based approach is a method to organize domain knowledge and to create a domain dictionary. PAORE provides a practical method to use packages themselves as a knowledge source and to create a domain dictionary.

PSM is one of the key knowledge in PAORE. PSM is generated from the analysis of packages and components in a specific domain from the viewpoints of features. Some domain analysis methods based upon features have been proposed. FODA includes a method to identify the distinctive features of a class of systems[5]. This kind of domain analysis methods may support the PSM generation in PAORE. Furthermore, some methods to describe domain knowledge in elements and patterns of workflow, e.g. work by Rolland et al.[12], have been proposed. We usually describe a workflow in common vocabularies and their hierarchical structure. We will take knowledge with procedural information like workflow into a consideration.

7 Conclusions

In this paper, we have proposed a new knowledge model of expert analysts based on the structure of package specification. Future research is needed to address the following issues:

1. PSM development environment should provide the following facilities:
   - Inquiry terms to describe functional and non-functional requirements.
   - Manage PSM information for revision and maintenance.
   - Investigate meta-knowledge used by experienced analysts in the course of requirements analysis.
2. In case of business application systems, it is necessary to clarify the relationship between PSM and the business modeling information that defines the context of the business applications.
3. It is needed to support negotiation process with clients.
4. The way how to estimate the efforts of requirements analysis using the proposed method is also needed.

We will also have experiments to investigate and evaluate the effectiveness of the proposed method.

Acknowledgments

This work has been supported by Grant-in-Aid for Scientific Research (C)(1) (KAKENHI #15500019), the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

[1] R. P. Diaz and G. Arango. *Domain Analysis and Software Systems Modeling*. IEEE Computer Society Press, 1991.

[2] X. Franch and J. P. Carvallo. A Quality-Model-Based Approach for Describing and Evaluating Software Package. In *Proc. of IEEE Joint International Conference on Requirements Engineering (RE 02)*, pages 104–111, 2002.

[3] R. L. Glass. The Naturalness of object orientation beating a dead horse? *IEEE Software*, 2002(May/June), 2002.

[4] Junzo Kato et al. A Model for Navigating Interview Processes in Requirements Elicitation. In *Proc. of APSEC 2001*, pages 141–148, 2001.

[5] K. Kang et al. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, ADA 235785), Pittsburgh, PA, 1990.

[6] J. Kontio. OTSO: A Systematic Process for Reusable Software Component Selection. Technical Report CS-TR-3478, University of Maryland, College Park, MD, 1995.

[7] A. Liu. Gathering Middleware Requirements. In *Proc. of 15th International Conference on Information Networking (ICOIN '01)*, pages 81–86, 2001.

[8] N. A. M. Maiden, C. Ncube, and A. Moore. Lessons Learned During Requirements Acquisition for COTS System. *Communication of the ACM*, 40(12):21–25, 1997.

[9] M. Mosko, H. Jiang, A. Samanta, and L. Werner. COTS Software Acquisition Meta-Model. In *Proc. of COTS Workshop*, 2000.

[10] C. Ncube and N. A. M. Maiden. Guiding Parallel Requirements Acquisition and COTS Software Selection. In *Proc. of 4th International Symposium on Requirements Engineering (RE'99)*, pages 133–140, 1999.

[11] M. Ochs, D. Pfahl, G. Chrobok-Diening, and B. Northhelfer-Kolb. A COTS Acquisition Process: Definition and Application Experience. In *Proc. of ESCOM-SCOPE 2000*, pages 335–343, 2000.

[12] C. Rolland and N. Prakash. Matching ERP System Functionality to Customer Requirements. In *Proc. of 5th IEEE International Symposium on Requirements Engineering (RE 01)*, pages 66–75, 2001.

[13] H. Rubenstein and R. Waters. The Requirements Apprentice: Automated Assistance for Requirements Acquisition. *IEEE Trans. on Software Engineering*, 17(3):226–240, 1991.