5th World Congress for Software Quality – Shanghai, China – November 2011

Utilization of Domain-Specific Knowledge for Quality Software Design

Noriko Iizumi
Hitachi
High-Technologies
Ibaraki, Japan
iizumi-noriko
@naka.hitachi-hitec.com

Tomoko Tomiyama
Hitachi
High-Technologies
Ibaraki, Japan
tomiyama-tomoko
@naka.hitachi-hitec.com

Atsuko Koizumi
Hitachi, Ltd.
Kanagawa, Japan
atsuko.koizumi.zt
@hitachi.com

Atsushi Takahashi
Hitachi
High-Technologies
Ibaraki, Japan
takahashi-atsushi
@naka.hitachi-hitec.com

Abstract

Domain-specific knowledge is highly important in embedded system development. In order to design software based on an appropriate rationale, engineers have to understand this domain-specific knowledge well. Usually, domain-specific knowledge is accumulated in individuals through experience, and there is no opportunity for sharing. This makes utilizing this valuable information in a team difficult. This paper suggests a practical method in which existing documents are used as a knowledge framework, for extracting and describing domain-specific knowledge. The use of this method allows for the creation of a questionnaire that extracts the necessary knowledge from specialists without introducing any notable increase in their workload. The extracted knowledge is described by two types of documents: general information manuals that explain using text, and a domain-specific knowledge model that shows the scope and relationships of the knowledge using a diagram. Domain-specific knowledge was extracted and shared by applying this method to five teams. The effect on the quality of software design was examined by causal analysis of past design faults. The results showed that 41% of the past faults would be prevented when utilizing the extracted domain-specific knowledge.

1. Introduction

In software development, the quantity of domain-specific knowledge as well as software development skill generally affects the software product's quality. When developing a series product, domain-specific knowledge, such as the user environment, the purpose of product installation, and cooperation with other related products, is especially required for an appropriate design. Hitachi High-Technologies develops and offers series products: semiconductor inspection equipment, DNA sequencers, and automatic clinical chemistry analyzers. Therefore, domain-specific knowledge is also required to design these products based on an appropriate rationale.

Although the importance of domain-specific knowledge is recognized, it is often not transferred adequately. There are probably two reasons for this. One is that specialists are not always aware of all the valuable knowledge they have obtained organically through experience. Although there are some studies of sharing domain-specific knowledge [1] and of a support system that facilitates communication [2], they are not used in actual practice. Another reason for the inadequate transfer of domain-specific knowledge is that the method to transfer domain-specific knowledge depends on the people involved. On-the-job training (OJT) is one approach to transfer knowledge or skill from person to person. However, in the current situation, where global outsourcing has recently emerged and many diverse members have been engaged in development, OJT is extremely difficult.

The primary purposes of this study are to determine the valuable knowledge held by specialists, to extract comprehensive and unbiased knowledge from specialists, and to describe the extracted knowledge so that it may be transferred without misunderstanding. In this paper, a practical method for the above is suggested with an actual application result. This chapter describes the background and purpose of the present study. In Chapter 2, concerns regarding the quality of software design are indicated, and related studies are reviewed. Chapter 3 describes the method for extracting and describing domain-specific knowledge. Chapter 4 presents the application results and the effect on the quality of software design. Chapter 5 provides a discussion of the method and its application. Chapter 6 presents a summary of the findings.

2. Deterioration in Software Design Quality Caused by Lack of Domain-Specific Knowledge

In this chapter, the relationship between domain-specific knowledge and the quality of software design is clarified. In addition, conventional methods to transfer domain-specific knowledge and preceding studies are reviewed. Then, the importance of sharing domain-specific knowledge widely is discussed, and the question addressed in the present study is defined.

2.1 Concerns Regarding Quality of Software Design

Development of a series product has frequently been seen in embedded systems. In fact, our flagship products undergo series development and are developed over ten years or more. In this situation, engineers have been engaged in development of the same product for a long time. They accumulate partial domain-specific knowledge while managing part of the system. As this activity is repeated, awareness of the importance of transferring knowledge that only a particular engineer possesses is lost.

Recently, with improvements in hardware performance, embedded system software has grown in size to more than one million lines of code. Accordingly, the composition of software development teams has changed from a few select members to many diverse members. Thus, the scope of development that an individual can experience has become limited. This tendency also limits their opportunities to acquire knowledge through work.

Development of large-scale software by many diverse engineers causes some difficulties regarding the quality of software design. Engineers with little experience of the target product interpret demands and design specifications in the light of their limited knowledge. Since the engineers may understand neither the demand nor the intention behind a specification correctly, design faults are generated. It is difficult to find this type of design fault only by testing. Therefore, the quantity of domain-specific knowledge affects the quality of software design in large-scale software development by many diverse engineers.

2.2 Conventional Methods for Transferring Domain-Specific Knowledge and Previous Studies

To date, concern has focused on tacit knowledge. The conventional methods for transferring domain-specific knowledge and previous studies on knowledge sharing are outlined below, and an unsolved problem is clarified.

(1) Conventional methods to transfer domain-specific knowledge

There are two approaches to transfer experiential knowledge, the personalization strategy and the coding strategy [3]. In the personalization strategy, knowledge is transferred from person to person. This approach is based on the idea that experiential knowledge accumulates only in a person who learned it through experience. OJT is a typical example. Although OJT used to be effective, knowledge cannot be transferred only by OJT as large-scale software development by many diverse engineers increases globally. In the coding strategy, knowledge is transferred by extraction and coding. This approach is based on the view that all the engineers can transfer experiential knowledge by enabling easy access to it in a database. This is also called the person-to-documents approach. A checklist is a typical example. A checklist is a document that transfers knowledge of countermeasures obtained from previous faults as lessons or know-how. Although a checklist is quite effective as a prescription, the effect is lessened if there is no corresponding description.

(2) Previous studies on knowledge utilization

Studies of knowledge management, which changes individual knowledge to valuable systematic knowledge, have been conducted since 1990. Their purpose is to enable the use of individual knowledge by anyone at any time. Moreover, studies on domain analysis have been conducted since the mid-1980s, at the same time as the modeling that suggested software reuse. Domain analysis is an attempt to identify the objects or operations that an expert thinks are important, as well as the relationships between them [4]. The

main problems in domain analysis are knowledge representation and knowledge acquisition [5]. Various expressive forms have been suggested for knowledge representation. However, knowledge acquisition has not been examined sufficiently.

Among the conventional method and previous studies, little work has been done concerning how to define and extract valuable knowledge from individuals. The important reasons for this are that the knowledge source is a person and the knowledge cannot be seen from the outside. A person who holds valuable knowledge often does not know how to offer or demonstrate it. On the other hand, those who are eager to obtain knowledge cannot explain what knowledge is necessary, because they do not grasp the whole picture.

2.3 Problem to Be Solved

When software is developed by many diverse engineers, almost all engineers must work with only limited knowledge. If all the engineers understand the relevant domain-specific knowledge, they can achieve a high-quality design based on an appropriate rationale.

In this paper, the valuable knowledge that should be shared is defined as "the domain-specific knowledge and rationales that must be understood when designing software." In an effort to complement engineers' limited domain-specific knowledge, two approaches are suggested. One is a method for extracting accumulated knowledge from specialists using existing design documents as a source of knowledge. Another is a method for describing domain-specific knowledge so that engineers can understand it well and apply it to actual software design. When domain-specific knowledge is organized and the engineers understand the rationale behind the product, the quality of software design will improve.

3. Method for Extracting and Describing Domain-Specific Knowledge

This chapter introduces a practical method for extracting and describing knowledge. First, the knowledge to be extracted is defined. Next, the method for extracting the target knowledge is shown. This method uses existing documents as a knowledge framework. Finally, a description of the extracted knowledge is given.

3.1 Definition of Knowledge That Should Be Extracted for Use in Software Design

Practical knowledge and theoretical knowledge are obtained during software development. Practical knowledge is understood through experience and includes not only know-how but also mental models and beliefs. Theoretical knowledge does not require experience and can be realized from books, magazines, and other sources. This study treats both types of knowledge, which are expressed verbally in order to transfer comprehensive content widely. The content of the extracted knowledge is the rationale behind the product, including the historical background. The tacit knowledge related to the software quality is also included. This kind of knowledge is defined as follows.

For understanding domain:

- The user's business
- The purpose of installing the equipment
- The user's operation flow
- Social motivations such as laws and regulations

For quality software design

- The historical reasons for changes of requirements or demands
- Background of the hardware technology
- Background of the software technology

3.2 How to Extract Domain-Specific Knowledge

There are three features in the suggested method: formation of a team, a framework of domain-specific knowledge based on existing documents, and an interview questionnaire. This method is designed to be conducted by engineers, because the extracted knowledge should be utilized by engineers themselves.

First of all, a team of two or more engineers, including one or more specialists who are likely to have experiential knowledge, is formed. At least two people are chosen to be a writer and reviewer, who maintain the objectivity of the review. For example, a software engineer might be assigned to be a writer, and a specialist might be assigned to be a reviewer as well as a knowledge provider.

Next, the source of domain-specific knowledge is selected in order to form the basis of the knowledge framework. Operation manuals, user requirements, and design specifications are suitable sources of knowledge. Since common and particular knowledge should be distinguished in series product development, the design specifications of two or more products in the series are referred to.

Then domain-specific knowledge is extracted by the following steps.

Step 1: Knowledge determination

> The writer provisionally determines the scope of the knowledge using the table of contents of the existing design specifications. The writer considers the items to extract and makes a table of contents for a new document, a general information manual explained in section 3.3 (1). It is essential to consider not the structure of a function, but the structure of the knowledge.

Step 2: Information collection

> From the design specifications of two or more products in the series, the writer looks for descriptions corresponding to the domain-specific knowledge. The writer then makes a knowledge framework by copying and pasting them into the new document.

Step 3: Information arrangement and question compilation

> The information pasted into the new document may itself be domain-specific knowledge, or it may be a key to domain-specific knowledge. Thus, the writer should arrange it carefully and generate a questionnaire. This questionnaire becomes an entry point to knowledge extraction from a specialist. To avoid any bias in the writer's thinking generated by his or her experience, the writer must list all the things about which he or she has doubts.

Step 4: Knowledge extraction from specialists using interview questionnaire

> Using the questionnaire created in step 3, the writer extracts knowledge from a specialist. Since a specialist's reply is often premised on different background knowledge, the writer needs to ask questions that reveal the premise. In addition, the writer discerns how detailed the information must be. This helps avoiding bias in the information and reduces the specialist's high workload.

Step 5: Knowledge description

> Based on the collected information, the writer arranges and documents the domain-specific knowledge. The new documents describing the domain-specific knowledge are a general information manual and a domain-specific knowledge model explained in sections 3.3 (1) and (2). The new documents are finally reviewed by the specialist.

By following the above steps, which reduce the specialist's workload, the background knowledge and rationale missing from the existing documents can be clarified. Furthermore, when others ask questions, the importance of knowledge that is obvious to the specialist becomes clear. This is an efficient and widely applicable method for extracting comprehensive domain-specific knowledge.

3.3 How to Describe Domain-Specific Knowledge

The extracted domain-specific knowledge is utilized to complement the engineers' limited knowledge. Considering the high resource mobility in recent software development, it is necessary to enable the study

of domain-specific knowledge at any time by a newly arrived engineer. Thus, the extracted knowledge is expressed as a general information manual using text and as a domain-specific knowledge model using a diagram.

(1) General information manual

The extracted knowledge is expressed as a general information manual. This document explains the purpose and the environment in which the product is used, so the engineer can understand the rationale behind the product. If all engineers can understand the background and the rationale for why the software exists and why it must perform certain functions, requirements can be appropriately met, and better design judgment can be achieved. The general information manual is written and reviewed with attention to the following points.

- Order of the table of contents.
  It is not suitable to compose the table of contents on the basis of functionality. Since the table of contents should make it easy to find knowledge required by new engineers, definitions of terms are given first, and the user's operation flow is described next.
- Description of the purpose.
  It is good to think about the user's viewpoint to understand why the product must behave in a certain way and why a function is required. The subject of design specifications is generally the product, and its behavior and functions are described. In contrast, the general information manual explains the user's operation flow, the user's purpose, and related matters. This increases awareness of unconscious premises and the background of the product.
- Lead sentences.
  A lead sentence is inserted at the beginning of each chapter or paragraph in order to clarify the point to be explained. Since text may be interpreted differently by different readers, it is effective to provide the information that the writer wants to transfer in a lead sentence at the beginning of each chapter.
- Description of the information common to products in a series.
  Whether the information is a basic matter common to all products in a series or whether it is a specific matter for only a particular product should be considered carefully. Mainly, basic matters common to products in a series are described in the general information manual.
- Review of the following

  - Is the manual arranged in order from the introduction to detailed arguments?
  - Are the need for the function and the premise described?
  - Is common knowledge described for series products?
  - Are terms defined and used consistently?
  - Are terms used with awareness of the appropriate subject, information, or role?

(2) Domain-specific knowledge model

Using the general information manual described above, new engineers begin to partially understand the domain-specific knowledge. However, the scope of domain-specific knowledge and the relationships between sections are difficult for new engineers to grasp thoroughly. Thus, a domain-specific knowledge model is created on the basis of the general information manual to clarify the relationships between the sections. A domain-specific knowledge model expresses relationships using nodes and arrows. A node indicates knowledge or a basic concept, and a line indicates the relationship between them.

Figure 1 shows an example of a domain-specific knowledge model. The dotted lines represent the boundaries between sections. Lines that cross these

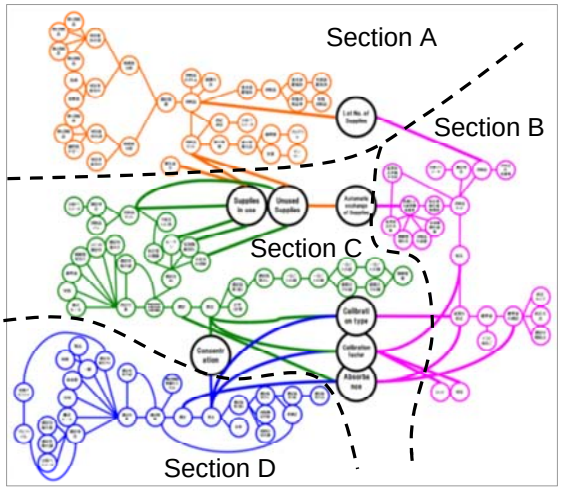

Figure 1. Example of domain-specific knowledge model.

boundaries represent knowledge relevant to both sections. Representing the knowledge as a diagram enables engineers to grasp the scope of the target domain and see what knowledge is shared between sections.

4. Results

This chapter presents an application of the suggested method and the utilization of the extracted domain-specific knowledge. The effect of this method on software quality is also examined.

4.1 Application of the Method

The goal of the suggested method is to utilize domain-specific knowledge for software design. To confirm the practical applicability of this method, it was applied in five teams consisting of engineers engaged in software development for a series product.

(1) Preparation

An automatic clinical chemistry analyzer was chosen as the focus of the domain-specific knowledge. The required knowledge was the user's workflow and fundamental knowledge regarding an analyzer, such as its purpose and the principle of the analysis. Thus, the knowledge to be extracted is defined as the fundamental content common to series products (general knowledge). It is focused on the content useful for new engineers. Five teams were formed to cover every section of the target domain. Thirty-nine engineers each chose a section of interest and joined the appropriate team. Each team had from 5 to 13 people. A leader, an assistant leader, and advisers (specialists) were placed on each team. The software development experience of team members varied from 1 to 26 years. The source of domain-specific knowledge was the design specifications of four or five distinct series products.

(2) Implementation

This activity was performed in a top-down manner as part of the team’s duties. The steps of implementation are as follows.

1) Explain the meaning of domain-specific knowledge extraction to all participants (manager).
2) Indicate the working hours available for domain-specific knowledge extraction (manager).
3) Organize teams and determine team leaders (extraction team).
4) Explain the domain-specific knowledge extraction process (support team).
5) Devise a domain-specific knowledge extraction schedule (extraction team).
6) Conduct domain-specific knowledge extraction (extraction team and support team).

(3) Actual results

The teams worked from May to September of 2010. The activity was promoted so that knowledge extraction would be completed in September. The support team encouraged the extraction team when deadlock occurred. The support team also provided rules for text creation in order to unify each writer’s style and improve readability, as well as to reduce wasted effort. The results of this activity are shown in Table 1.

Table 1. Results of general information manual creation

SECTION (TEAM)	NUMBER OF TEAM MEMBERS	NUMBER OF MEETINGS	NUMBER OF PRODUCTS REFERRED TO	TOTAL PAGES OF SOURCE DOCUMENTS	TOTAL PAGES OF OUTPUT
A	8	13	5	1123	32
B	13	14	4	616	51
C	7	17	4	493	31
D	5	15	4	154	36
E	6	12	4	1617	33

4.2 Utilization of Extracted Domain-Specific Knowledge

The extracted domain-specific knowledge described in the general information manual and the domain-specific knowledge model are shared among engineers who are new to a project and engineers

with limited knowledge of the target domain. Two methods to transfer domain-specific knowledge were used: lectures and self-education.

(1) Lecture system

The general information manual was used in the initial training for software engineers. A total of five candidates were trained: two with no experience in software development, two with less than three years of experience, and one with more than three years but less than ten years of experience. The three engineers with experience in software development had limited domain-specific knowledge. The lecturer explained the general information manual after the engineers had studied it for two hours. After the lecture, the engineers' understanding was surveyed using a questionnaire. The responses shown in Figure 2 indicate that almost all of the engineers understood the material, and their understanding was mostly independent of the level of software development experience.

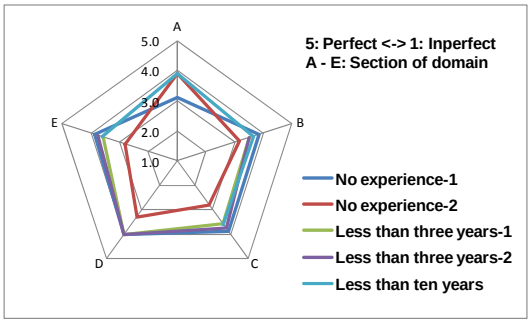

Figure 2. Degree of understanding.

(2) Self-education system

The five general information manuals were saved on a server so that engineers who were not involved in their creation could also access them. New engineers were asked to read the manual beginning in October 2010. With self-education, an overview of the domain-specific knowledge can be grasped in one to two days. In an unexpected result, people in other product departments used this resource. Furthermore, people use the general information manual and domain-specific model as a checking tool to confirm the scope of their knowledge.

4.3 Examination of the Effect on the Quality of Software Design

The purpose of this study was to prevent the deterioration in software quality caused by lack of experiential knowledge. Of course, the extracted domain-specific knowledge has already been provided to the engineers. However, to check whether software quality has improved as intended because the engineers complemented their domain-specific knowledge, development must be tracked for several years. As an alternative method, a causal analysis of previous faults was used to examine how many faults would have been eliminated.

This examination was conducted for one of the five sections for which information was extracted; there were 39 faults in this section (Figure 3). Eighteen were relevant to domain-specific knowledge, and 11 of those could have been prevented if engineers understood the contents of the general information manual. Five other faults required knowledge shared between sections. Thus, these faults could also have been prevented if engineers understood the relationships between sections given by the domain-specific knowledge model. A total of 41% of the faults (= (11 + 5)/39) could have been prevented using extracted domain-specific knowledge.

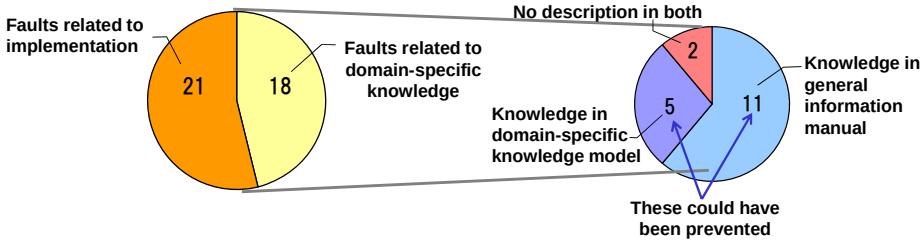

Figure 3. Classification of past faults.

5. Discussion

In this chapter, the availability of the suggested method, an evaluation of this application, and the utilization of tacit knowledge are discussed on the basis of the results of the application of the proposed method.

(1) Availability of the suggested method

Engineers who had less than one year of software development experience were unskilled at extracting basic information related to domain-specific knowledge from the existing design specifications. They could not distinguish whether something was background or only a specification, because they had relatively little understanding of software development practice. Engineers with three years or more of software development experience could distinguish domain-specific knowledge needed for software development even if the domain varied. On the other hand, it became clear that the specialists were not always confident about the knowledge they accumulated experientially. Creating the output using teamwork solves these problems.

(2) Evaluation of this attempt

In fact, there were already some documents made by veteran engineers explaining details of domains. However they have not been utilized, because there were various intentions behind their creation or they had only partially complete contents. The suggested method has a clear step defining domain-specific knowledge and its scope. Future work will address the maintenance of the contents. A periodic review is planned based on when new engineer assignments occur.

(3) Utilization of tacit knowledge

The general information manual and domain-specific knowledge model made it easy for new engineers to acquire knowledge. Until now, engineers have understood domain-specific knowledge primarily through experience. Alternatively, engineers had to obtain the necessary information by simply reading the existing specification or finding it directly from a specialist. The opportunity cost model reveals that this method is extremely effective for both engineers and specialists.

6. Conclusions

The importance of utilizing domain-specific knowledge in large-scale software development by many diverse engineers is unquestionable. This paper suggested a practical method for extracting and describing domain-specific knowledge. Using existing documents as a knowledge framework and a questionnaire as the basis of knowledge extraction is a key feature ensuring the completeness of the knowledge and reducing the specialist's workload. This method also enables the extraction and description of domain-specific knowledge by engineers who are unfamiliar with the domain. Domain-specific knowledge is described in two types of document: the general information manual using text and the domain-specific knowledge model using a diagram. Using these documents, engineers can acquire domain-specific knowledge with self-education at any time.

In conclusion, the essential aim of this effort is to make every engineer notice the scope of his or her understanding of the domain. In this way, adaptable thinking and the habit of design based on an appropriate rationale can be mastered. This supports the organization's capability as well as the quality of its designs.

References

[1] Nakayama, Y., DFACE-KM Knowledge Management Methodology, Toshiba Review Vol.60 No.12 pp.48-49, 2005.
[2] Umeki, H. and M. Horikawa, Platform for Community-Based Collaborative Knowledge Creation, Toshiba Review Vol.56 No.5 pp.14-18, 2001.
[3] Davenport, T.H., Integrating Knowledge Management into the Organization, Diamond Harvard Business Review pp.26-36, September 1999.
[4] Neighbors, J.M., The Draco Approach to Constructing Software from Reusable Components, IEEE Transactions on Software Engineering Vol.10 No.5 pp.564-573, 1984.
[5] Prieto-Diaz, R., Domain Analysis: An Introduction, ACM SIGSOFT, Software Engineering Notes Vol. 15 No.2 pp.47-54, 1990.