A Case Study: Requirements Elicitation Processes throughout a Project

Takako Nakatani
University of Tsukuba
The Graduate School of Systems Management
3-29-1, Otsuka, Bunkyo-ku, Tokyo, 112-0012, Japan
nakatani@gssm.otsuka.tsukuba.ac.jp

Shouzo Hori
Yaskawa Information Systems Corporation
Quality Assurance Dept.
1-2-3, Manpuku-ji, Asou-ku, Kawasaki, Kanagawa, Japan
hori@YSKcomnet.co.jp

Naoyasu Ubayashi, Keiichi Katamine, and Masaaki Hashimoto
Kyushu Institute of Technology
Graduate School of Computer Science and Systems Engineering
680-4 Kawazu, Iizuka-shi, Fukuoka, 820-8502, Japan
{ubayashi, katamine, hashimoto}@ai.kyutech.ac.jp

Abstract

Requirements changes sometimes cause a project to fail. A lot of projects now follow incremental development processes so that new requirements and requirements changes can be incorporated as soon as possible. These processes are called integrated requirements processes which integrate requirements processes with other development processes. We have investigated the requirements processes of a project from beginning to end, both quantitatively and qualitatively. Our focus is to clarify the types of requirements based on the components contained within a certain portion of the software architecture. Each type reveals its typical requirements processes through its rationale. The case studied is a system to manage the orders and services of a restaurant. In this paper, we introduce the case and describe the types of requirements processes. Then we discuss the reasons why we could not elicit all the requirements in the early development processes.

1. Introduction

Requirements changes are considered detrimental to the success of a development project. Generally, however, requirements are changed. To deal with the nature of requirements, there is a proposal to integrate requirements processes with other development processes, such as design, implementation, and testing processes[8]. Our case study aims to analyze the requirements process after the requirements analysis phase has been completed. After studying project records both quantitatively and qualitatively, we have categorized the requirements process according to the components contained within a certain portion of the software architecture. Each category reveals its typical requirements processes through its rationale.

In Section 2 we give an overview of the project. Section 3 illustrates the results of our study and provides the requirements processes of the target project. In the next section, we categorize the requirements processes into six types, followed by a discussion of the characteristics of each. In Section 5, we introduce related works. In the final section, we present our conclusions.

2. Overview of the project

The target project was initiated to develop a restaurant service and order management system, named RESORT. RESORT can accept orders from the hand held terminals of staff members and from the table terminals. Table terminals are set on guest tables to provide the food menu, recipes, knowledge of liquor and sake, and entertainment information via a wireless network from the application server. A previous system was developed by a certain company several years ago. The client of this project decided to re-develop the system to improve its installation load and extensibility. The system was required to be able to replace its Or-
The project was carried out by one client and three developer companies. One of the developer companies provided table terminals with an OS and a browser for HTML. The second company developed contents for the table terminals. The third company, the YSKcom, developed an application server that could communicate with the other components. The YSKcom provided us with their development data. The client provided a shared repository for the project to manage the schedule and problems. The developers communicated with each other via the repository and solved any problems that occurred during the development. OES providers were not members of the project. The specifications of the OESs were made available to the project via the client.

The RESORT project was completed in four and a half months, although five months were originally budgeted. The development data were collected in 117 days, from the external design phase to completion of the project.

Before starting the project, the YSKcom proposed the software architecture illustrated in Figure 1 to solve the problems of the previous system. In the figure, the white icons represent components developed by the YSKcom, the dotted line rectangle represents the OES, and gray colored icons represent the components developed by other developers. Our research focuses on the components in white. This architecture was well elaborated and shared with the developers in order to understand the functions of the components.

The duration of each phase was accomplished as shown in Table 1. We counted the duration from the start of the external design phase. The duration of the integration testing phase was actually 27 days according to the recorded data. However, we counted it as seven days, since only four requirements were elicited in the first seven days of the phase. Thus, the project was completed on the 117th day.

3. Requirements process

3.1. A case study method

The engineers from the YSKcom used traditional interview techniques in multiple non-scheduled reviews with their client. They managed issue records, question and answer records, and decision records for each phase. From the records, we can see the content of each requirement, its priority, the date of contact/recording/completion, the name of a client/an engineer in charge/the manager, agreement of stakeholders, and the reason for the change/addition/deletion of a requirement. We analyzed these documents and counted the number of additional requirements, changed/adjusted requirements, and deleted requirements. Thus, if a newly added requirement was deleted after being changed, we counted it three times.

The quantitative data were analyzed in two ways: first, on a quality characteristics basis, and then, on an architectural basis. Furthermore, we interviewed both the project manager and the quality manager of the project to understand the background of the records and their rationale.

3.2. From a quality characteristics view

Figure 2 shows the requirements processes from a quality characteristics view. The x-axis represents elapsed days of the project, while the y-axis represents the cumulative number of elicited requirements. From this graph, we can see that the requirements of (1) functionality and (3) usability were elicited continuously throughout the project, while other requirements were frozen midway through the project schedule.
The continuous process for the functionality requirements represents the learning curve of the engineers, as they did not have sufficient knowledge to understand the domain of RESORT. They had to develop the system as they acquired the knowledge in a stepwise manner. Furthermore, the OES providers did not make their specifications available to the project in the early phase of development. The major provider only released his specification to the project after the implementation phase started, i.e., on the 64th day of the project schedule.

The continuous process for the usability requirements represents not only a process of trial and error to achieve better usability, but also a process of eliciting undefined requirements by evaluating the system adequacy. In fact, although the project was preceded by the waterfall model, the part relating to the usability or user interfaces was repeatedly developed and validated. If a usability specialist or domain specialist had been available to the project, the process might be represented by a different curve. The requirements elicited in the system testing phase were transferred to the next version of the system.

Regarding the non-functional requirements, these were defined as a solution to the problems of the previous system. The client recognized the quality problems of the previous system from their business objectives. Therefore, these processes could be frozen before the internal design started.

### 3.3. From the component view

The functionality and usability requirements depend on components. Figure 3 illustrates a requirements process for each component. In the figure, the x-axis represents elapsed days of the project, while the y-axis represents the cumulative number of elicited requirements. The results show that the Table Terminal Communicator (TT_C) and System Monitor (Sys_M) received many requirements throughout the project. These components are situated in the user interface area. The OES Communicator (OES_C) also shows changes to its requirements in the latter part of the development. This component depends on the specifications of the OES. The system was required to connect to multiple vendors’ OESs, the specifications of which were, in fact, provided after the implementation had started. Until then, the engineers had collected product information on an OES informally.

We focused on the requirements elicitation ratio according to the project schedule. Let \( r_i \) be the number of elicited requirements on project day \( i \). Thus, the sum of a set of elicited requirements in \( n \) days can be written \( \sum_{i=0}^{n} r_i \). The ratio of the accumulated requirements \( R_i \) on the \( i \)th day can be defined as

\[
R_i = \frac{\sum_{j=0}^{i} r_j}{N}
\]

Here, \( N \) is the sum of the number of requirements elicited throughout the project. Hence, \( R_0 \) is equal to 0% and \( R_{end} \) is equal to 100%.

Figure 4 shows the ratio of elicited requirements \( R_i \) versus the project schedule. The project schedule in
Figure 4. Requirements elicitation ratio

this figure represents total days. For example, the internal design phase was planned after the external design phase, which took 40 days. Its process appears after the external design phase in Figure 4. As a result, the total number of days of the project is shown as 154 days, rather than 117 days. The vertical line in Figure 4 represents the end of each phase.

To clarify the reason and rationale of the requirements processes, we conducted interviews with both the project manager and quality manager. Then, we analyzed the rationale of the requirements processes according to the components. The numbers in the following list correspond to the numbers shown in Figure 4.

- 1. Table Terminal Communicator ($TT_C$) and 2. System Monitor ($Sys_M$)
The requirements have been elicited continuously from an early phase to the end of the project. These components are in the user interface area. For example, $TT_C$ depends on the Table Terminal ($TT$) used by guests of the restaurant. The $Sys_M$ is for use by the manager of the restaurant. The usability of these components is the key issue in enabling RESORT to compete with similar products. Therefore, the engineers elicited requirements for these components repeatedly through discussions with the client to improve its usability and functionality.

- 3. Installation Support ($Inst_S$)
The client identified the problems of the previous system from a product business view before the project began. The requirements for the $Inst_S$ were defined as solutions to these problems. The client could therefore clearly define their requirements for the $Inst_S$ early in the development.

- 4. Table Terminal Monitor ($TT_M$) and 5. Center Communicator ($Cent_C$)
The requirements elicitation ratio reached 100% by the end of the external design phase. These components depend on external interfaces provided by the co-developers. The members of the development team highly prioritized the review of these specifications to discover incomplete requirements and could thus freeze the requirements in the early phase. They were able to collaborate well by sharing the problems of the project via the shared repository.

- 7. $DB$ The requirements elicitation ratio reached 100% by the end of the external design phase. The specification of this component was a reused one from the previous system.

- 6. OES Monitor ($OES_M$) and 8. OES Communicator ($OES_C$)
The requirements have been elicited continuously from an early phase to the end of the unit testing phase. These components depend on other components provided by outside companies. Furthermore, two developers with insufficient knowledge of OESs were assigned the task of managing integration of the system with respect to multiple OESs. Given this situation, the developers decided to study the OES domain based on the most popular OES product. After designing the component to fulfill these specifications, they then proceeded to adopt other OES products and change the design. Because of resource problems, it was difficult for them to fulfill every OES specification at one time.

For the $OES_M$, the requirements elicitation ratio reached 100% in the implementation and unit testing phase. Such requirements changes can be detrimental to the project schedule. Fortunately, the $OES_M$ had fewer functions and the change to the requirements was the deletion of functions developed for the most popular OES, although this caused several inconsistencies in the system.
4. Discussion

In the real world, a requirements process expresses mixed processes, since a requirement is constrained by an architectural aspect, as well as the context of the project. The $OES_C$ and $OES_M$ are examples of this. They are affected by the level of cooperation of the OES companies, the engineers’ knowledge, and the limitation of project resources, as well as the client’s business strategy. The client believed that $RESORT$ should be able to integrate with most available OESs to compete with similar products. In defining the requirements process model, we decompose the requirements process characteristics and simplify realistic situations.

Figure 5 depicts graphs of each type of requirements process. The $x$-axis represents the elapsed time of the project, while the $y$-axis represents the requirements elicitation ratio. We define each type as follows:

- **Reused type ($Type_{Ru}$)**
  An example of this type observed in the case study is $DB$. $Type_{Ru}$ is independent of other components and is selected and changed based on requirements. Therefore, requirements for $Type_{Ru}$ can be defined in an early development phase.

- **External interface dependent type ($Type_{Xi}$)**
  This type is for those components that depend on external components. Basically, external interfaces can be defined in the early development phase if their specifications are available. $TT_{M}$ and $Cent_{C}$ are happy examples of this type, in that the developer successfully procured their specifications in the early development phase. $OES_{M}$ and $OES_{C}$ are the unhappy examples, since their specifications were not available until the implementation had started. Thus, if the vendor of an external component cooperates with the project, its requirements can be frozen in the early development phases. On the other hand, if a vendor does not cooperate with the project, the situation can be made worse. These cases are shown as $Type_{Xi}$ and $Type_{Xi}'$, respectively, in Figure 5. The $Type_{Xi}$ needs the cooperation of related component vendors[5].

- **Diversity type ($Type_{DiV}$)**
  This type needs an appropriate design to deal with the variability of components. The developers can design adequately, if, and only if they have enough knowledge of the variations[6]. An example of this type observed in the case study is $OES_{C}$. The requirements for this were elicited continuously, because of the lack of knowledge of the developers.

- **User interface dependent type ($Type_{Ui}$)**
  This type is for a component that is dependent on a user interface. Requirements for these components tend to be added and/or changed even in the validation testing process, and sometimes in the maintenance phase as well. Such a component, therefore, requires incremental development by prototyping[3, 9]. Examples of this type observed in the case study are $Sys_{M}$, $TT_{C}$, $TT_{M}$.

- **Business requirements dependent type ($Type_{Bz}$)**
  This type is for a component that depends on a business strategy. We have divided this type into two subtypes. $Type_{Bz_{1}}$ relates to business improvement. Requirements for this type can be clearly defined before the development starts, because such requirements motivate the client to initiate the project. An example of this is observed in the requirements process of the $Inst_{S}$. $Type_{Bz_{2}}$ relates to a competitive business situation. We can see examples of this in $Sys_{M}$ and $TT_{C}$. To define the requirements for $Type_{Bz_{2}}$, we have to develop prototypes to validate the requirements in order to compete other products. For the $Type_{Bz_{2}}$, the client have to cooperate closely with the developers[4].

New requirements or requirements changes sometimes delay the project schedule. To save our project, we must identify the influential components that form the architecture, as well as plan and monitor the requirements processes of these components carefully. In our future work, we intend studying other cases to evaluate the adequacy and/or inadequacies of these types.

![Figure 5. Types of requirements and their observed processes](image-url)
5. Related work

The requirements process during the development has been extensively researched in the literature.

Aoyama and his colleagues studied the kinds of requirements that were changed or added after the requirements analysis phase. Then, they introduced an interview guide to clarify unstable requirements and evaluated the guide to elicit those requirements completely before the design phase started[1]. We do not think that it is possible to elicit requirements completely in the requirements phase, because our clients sometimes do not know their requirements. Our approach is that the engineers ascertain requirements together with their clients during the system development. To do this, we must manage the requirements process according to the type of each component and the project situation.

Arkley and his colleagues described an application of traceability by a company. They classified the project requirements and showed requirements changes in the development process. In their article, the specification and requirements were frozen before the software design review[2]. Sankar and Venkat focused on a way to control requirements. They showed the percentage of requirements frozen in the development process. According to the article, 70% of all requirements were frozen during the requirements gathering[7]. If most requirements could be frozen in the early development phase, we would be happy. One of the real problems is that many requirements sometimes remain undefined until the design phase. We studied a case in which the engineers elicited requirements throughout the development to clarify the practical situation. In practice, the client cannot show all the requirements before the design phase has started.

Houdek and Pohl studied the requirements engineering process of Daimler Chrysler. They mentioned that 50% to 60% of requirements changes are in the interface area[5]. We categorized this type of component as TypeUi. They also mentioned that the requirements engineering activities are heavily intertwined. Sommerville referred to the intertwined requirements engineering process as integrated requirements engineering[8]. From our study, we can conclude that an integrated requirements engineering process needs requirements elicitation scheduling techniques, as well as elicited requirements management.

6. Conclusions

We studied the requirements processes of RESORT. From its development records, we could categorize the requirements processes into seven types. These types have been observed from a type specific requirements process view. Our aim was to identify the type of each component in order to plan the requirements processes in the project. The results imply that we would be able to schedule the requirements process according to the type of each component and the project situation.

Acknowledgement

This work was funded by the Joint Forum for Strategic Software Research (SSR). We would like to thank our colleagues at SSR. We would also like to thank Mr. Masao Shimoji and his colleagues at the Yaskawa Information Systems Corporation who cooperated on our study as interviewees.

References