TOWARDS A STRATEGIC REQUIREMENTS ELICITATION
A proposal of the PRINCE Model

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

Shouzo Hori,
Yaskawa Information Systems Corporation,
1-2-3, Manpukuji, Asou, Kawasaki, Kanagawa, Japan
hori@ysknet.co.jp

Michio Tsuda,
Osaka University,
1-5, Yamadaoka, Suita, Osaka, Japan
michio.tsuda@easesoken.com

Mari Inoki,
Toshiba Solutions Corporation,
3-22, Katamachi, Fuchu-shi, Tokyo, Japan
Inoki.Mari@toshiba-sol.co.jp

Keiichi Katamine, Masaaki Hashimoto
Kyushu Institute of Technology,
2-4 Hibikino, Wakamatsu-ku, Kitakyushu-shi, Fukuoka, Japan
{*katamine, hasimoto*}*@ai.kyutech.ac.jp*

Keywords: requirements engineering, requirements elicitation process, case study.

Abstract: Requirements changes are sometimes pointed out as being one of the causes for project failure. Our solution to cope with this problem is to plan and manage requirements changes strategically. This paper introduces the PRINCE model that consists of 3+1 types of requirements elicitation processes based on the time of the maturation of the requirements elicitation activity. To explain the model, we show a real case with quantitative observations of a requirements elicitation process. When we are able to elaborate a strategy of requirements elicitation with the PRINCE model, we can elicit requirements by need of the physical development, rather than the theoretical development process model.

1 INTRODUCTION

Requirements changes are sometimes pointed out as being one of the causes for project failure. Various solutions exist to cope with this problem. For example, Trawling in *Volere* (Robertson and Robertson, 1999) is a method for requirements elicitation without possible omissions. Goal-oriented analysis methods explore the "why" aspect of requirements in order to find alternative requirements and forecast variability of requirements (Anton, 1996; Dardenne et al., 1993). Even though we apply those methods, it is hard to elicit requirements completely in the early stage of software development. Project managers must plan a sound development process that fits the fragility of the requirements with regard to each project situation.

There are several development process models that cope with requirements changes. An agile development process (Beck and et al., 2001) recommends incremental development in short durations. If the development duration is short, the project may be free from requirements changes caused by business environmental changes. The Unified Process (Jacobson et al., 1999) proposes a four-stage development. It focuses on a framework that takes measures to meet the future requirements changes. If the requirements change frequently, and they *are* a hazard for the project, we must consider a plan for the requirements elicitation throughout the whole development phase. Planning the requirements elicitation seems to be a realistic way of thinking, as opposed to *setting* the requirements process in the early development stage.

A model called the PRINCE (Pre Requirements Intelligence Net Consideration and Evaluation) model was developed in order to emphasize the requirements elicitation process and/or plan the requirements process. The PRINCE model assumes that requirements are elicited continuously during the whole development process. Sommerville referred to such a concept as Integrated Requirements Engineering (Sommerville, 2005).

This paper is constructed as follows. In Section 2, we discuss related works. Section 3 provides the PRINCE model and its four process types of requirements elicitation. In Section 4, we introduce a case

as an example project of the PRINCE model and analyze the project situation by mapping in accordance with the PRINCE model. In the final section, we conclude with the expectations and possible problems of the PRINCE model.

2 RELATED WORK

The requirements process during software development has been extensively researched in numerous volumes.

Lehman proposed the laws of program evolution (Lehman, 1978; Lehman, 1996). The first law "continuing change" said that "software systems that solve a problem or implement a computer application in the real world must be continually adapted for satisfaction." He focused on the software maintenance process. We focus on a software development process. As the development progresses, the customers understanding of the system deepens. Hence, the customers cannot help changing their requirements (Nakatani et al., 2008). The first law is applicable during the development process.

Nakamura analyzed project managers' efforts in dealing with specification changes (Nakamura, 2005). There are differences between new market projects and established market projects. The number of specification changes per MPI (Management Performance Index) is 0.116 and 0.01 for the new market projects and the established market projects respectively.

Finker and his colleagues clarified the handshaking process between a requirements provider and a solution provider (Fricker et al., 2007). Their research has been done for distributed product development and they have developed a technique to enable explicit *handshaking* procedures amongst stakeholders in order to deal with the risk of misunderstanding requirements. The handshaking process is composed of three activities: i.e. requirements communication, synthesis of the problem domain information/technology knowledge, and negotiation. They discussed implementation proposals that describe how a given requirement is intended to be realized by a software solution in order to validate the understanding of a requirement.

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 had started (Aoyama et al., 2007). 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 studied case 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 specifications and requirements were frozen before the software design review (Arkley and Riddle, 2006). 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 requirements gathering (Kousik and Raman, 2007). 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.

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 (Houdek and Pohl, 2000). They also mentioned that requirements engineering activities are heavily intertwined. Sommerville referred to the intertwined requirements engineering process as integrated requirements engineering (Sommerville, 2005). Our approach is to propose a process which plans and manages requirements elicitations throughout the project.

3 THE *PRINCE* MODEL

Figure 1 shows the PRINCE model. The model represents a requirements elicitation process with the ratio of requirements maturation versus the project schedule.

3.1 Ratio of Requirements Maturation

The ratio of requirements maturation represents how much the requirements are elicited for each software component, for example, physical software component, subsystem, a group of these components, quality characteristic, etc.

The ratio is defined as follows. 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 requirements maturation R_j is defined by the accumulated requirements on

the j th day as

$$R_j = \sum_{i=0}^{j} r_i / N$$

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%.

3.2 Maturation Types

We can observe the matured period for each software component. The three vertical lines in Figure 1 represent the end of the early, middle, and later developments stages respectively. The early development stage designates an integrated stage of requirements analysis and external design phase for water-fall process model, or an inception phase of the Unified Process. The middle stage designates an internal design phase, or elaboration phase. The later stage is corresponding to the phases following to the middle stage.

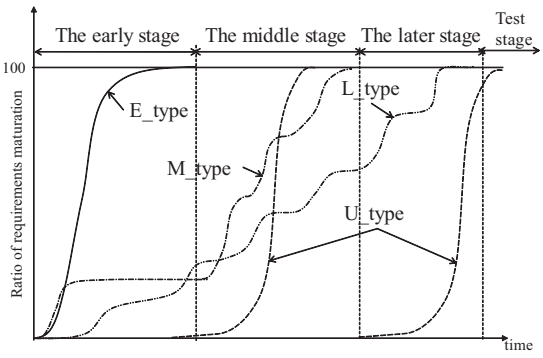

Figure 1: The PRINCE model

As shown in Figure 1, the PRINCE model consists of 3+1 maturation types of requirements elicitation process of software components. In the PRINCE model, the word "component" designates a physical software component, a group of them, or a subsystem. The types are the early stage maturation, the middle stage maturation, the later stage maturation, and the unforeseen maturation type. We named E_type, M_type, L_type, and U_type for each type respectively. The PRINCE model supports project managers in planning the requirements elicitation process strategically to fit the project situation. The concept of the model is, "elicit requirements by need for the development." The PRINCE model provides a plan of requirements elicitation to obtain 100% requirements by the end of the development. The characteristic of each type is as follows:

- E_type: The early stage maturation type
  When a requirements elicitation process follows the E_type, 100% of the requirements can be elicited by the end of the external design phase in the waterfall process model (WF) or the inception phase in the Unified Process (UP) and are never changed after this phase. These requirements will be designed and implemented after requirements have been elicited completely.
- M_type: The middle stage maturation type
  A requirements elicitation process follows the M_type, when the project cannot freeze the requirements until the implementation phase of the WF or the elaboration phase of the UP has started.
- L_type: The later stage maturation type
  If a requirements elicitation process follows the L_type, we have to elicit requirements even in the implementation phase. This means that incremental development is one of the recommended development processes for the type_L.
- U_type: The unforeseen maturation type
  This type of process cannot be planned. The unforeseen requirements are elicited unexpectedly during any of the phases. When we plan the requirements elicitation process, we must prepare for unforeseen requirements. We show two U_type processes in Figure 1 as examples.

3.3 Manage the Backlogs

To manage the requirements elicitation process, we need to watch two kinds of backlogs. One is for the requirements that have to be satisfied within the developing product, but are unforeseen in the early stage of the development. This type of requirements belongs to the U_type. The other is for the requirements that are defined to satisfy in the future product. This type of requirements elicitation process forms like an L_type. From a business perspective, eliciting such requirements is a benefit for the next project.

4 CASE OBSERVATION

We selected a five-month project in order to observe and analyze the requirements changes process by mapping in accordance with the PRINCE model.

4.1 The Case Overview

The target project was initiated to develop a restaurant service and order management system, named *RESORT*. *RESORT* accepts orders from the hand held terminals of staff members as well as from the table terminals.

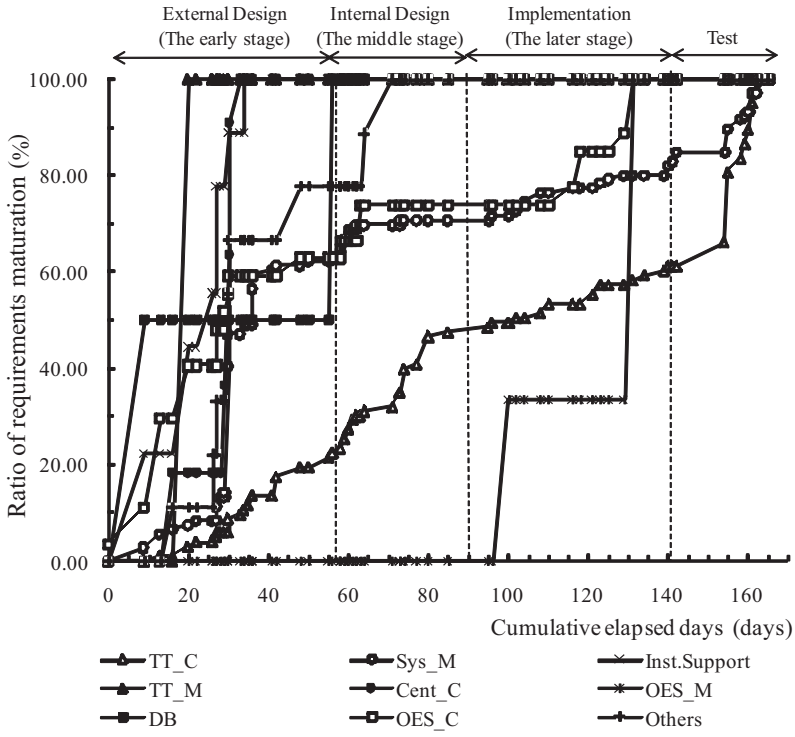

Figure 2: Requirements maturation ratio observed from a physical component view.

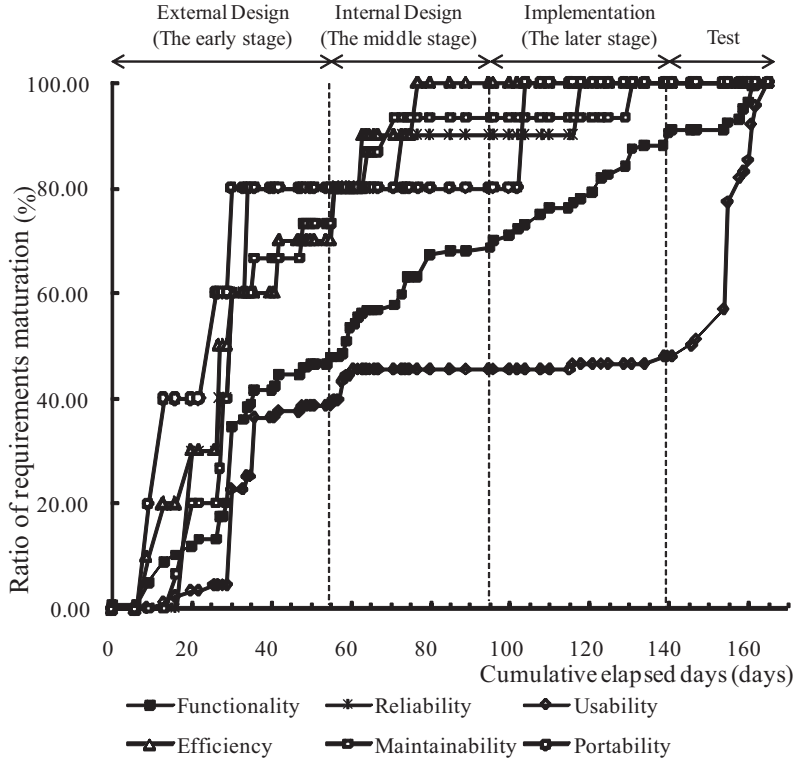

Figure 3: The requirements maturation rate of quality characteristics.

The *RESORT* project was completed in four and a half months, which fell within the planned schedule. The duration of the recorded data was 117 days. As shown in Table 1, records were kept for each development phase, i.e. an external design phase, an internal design phase, an implementation phase, and a testing and debugging phase. Every phase was overlapped with the other phases. The cumulative development duration was 165 days. Figure 2 shows the requirements maturation rate versus the cumulative development duration from a physical component view. Figure 3 represents the requirements maturation rate of quality characteristics.

Table 1: Duration of each phase.

Exter. Design	Inter. Design	Imple.	Testing	Total
55	40	45	25	165

4.2 Map to the *PRINCE* Model

We have mapped the requirements elicitation process as observed in the case in accordance with the types in the PRINCE model. We also interviewed the project manager to ascertain how he managed the requirements elicitation process.

- E_type:
  There are four Components, *Inst.Support*, *DB*, *TT_M*, and *Cent_C*, whose elicitation process follows the E_type. They matured in the early stage of the development shown in Figure 2.

  Inst.Support was required to improve the previous system. The client clearly recognized the problems of the previous system. Consequently, the project was able to completely acquire the requirements in the early stage of the project. The *DB* was a reused component. The client also clearly defined some minor changes for the existing component. Requirements for the other components were provided by the cooperating companies in the form of specifications of the external interfaces. These cooperating companies shared a common goal of completing the project within the schedule.

  According to the case, E_type can be applicable at least in the following situation.

  – The developer can contact a stakeholder who recognizes the problems of the former system.
  – There is a reusable component and its specification. Furthermore, the developer can access a stakeholder who understands the specific problems to be solved.
  – The developing system needs an external interface provided by the cooperative organizations and they share the goals of the project with the developers.

- M_type:
  Regardless of the situation, we have to plan the process for quality requirements as the M_type or the E_type, since they easily affect the software architecture. The process of eliciting efficiency requirements followed the M_type, as shown in Figure 3. The process of reliability, maintainability, and portability requirements did not follow M_type, because these requirements were related to the unexpected requirements elicitation in the later stage. We discuss the problem in the U_type explanation.

- L_type:
  OES_C, *Sys_M*, and *TT_C* followed the L_type. They matured in the later stage of the development, as shown in Figure 2. These components were developed incrementally. *OES_C* relates to the external interface of products provided by several third party companies.

  The project could not get the correct specifications in the early and the middle stages, because the third party companies did not cooperate with the project. Furthermore, the developers did not have enough knowledge of the problem domain. However, during the incremental development, they acquired the knowledge.

  The other components, *Sys_M* and *TT_C*, are related to the user interface, and played a key role in competing with other similar products on the market. According to this case, we have to consider the possibility of applying the L_type in the following situation.

  – The developers do not have enough knowledge of a portion of the system and need time to acquire the knowledge.
  – No usability specialist joins the project, and the project expects their customers to evaluate the UIs.
  – The components mentioned above are the competitive parts of the system, and the market situation is changing continuously.

- U_type:
  This type also appears in Figure 3. Requirements for *OES_M* were changed because of the inconsistency between the specifications and the actual product. It caused reconsideration of the quality requirements: reliability, maintainability, and portability.

5 CONCLUSIONS

One of the purposes of this paper is to present the PRINCE model as a method to plan the requirements elicitation process strategically. Before planning the requirements elicitation process, managers identify some sort of component as, "a preparation of requirements elicited" after the requirements analysis phase or the inception phase. The requirements elicitation process can be planned for each component according to the types in the PRINCE model. A use case, function, feature, quality characteristic or physical component are the candidate of the unit to be observed.

Throughout the project, the manager observes and evaluates the process of the *real* requirements elicitation process with two kinds of backlog for comparisons with the *planned* process. If we can elaborate a strategy of requirements elicitation with the PRINCE model, we will be able to apply different kinds of development processes within a single project.

The other purpose of this paper is to clarify a process of requirements elicitation quantitatively as the PRINCE model. The PRINCE model provides three types of requirements elicitation process, and one unforeseen type. Even for the unforeseen type, if we worry about the encounter with the U_type process, we have to prepare for accidental situations. Therefore, it is important to plan the requirements elicitation process.

The concept "eliciting requirements by need for the development" requires the manager to plan the future requirements elicitation according to the project situation. Still now, we cannot point out all possible situations for each type. We need to observe other requirements elicitation processes in real projects. Observation of projects provides us with a lot of knowledge to succeed in our project.

We are developing a guideline to solve problems through the observation of requirements quantitatively. For example, in this case, several requirements for the future product are included in current requirements. We need a way to distinguish these backlogs and the truly essential requirements for any specific project.

ACKNOWLEDGEMENTS

This project has been supported by Joint Forum for Strategic Software Research since 2007. We give thanks to all the members of this project and cooperators. The support of the engineers was essential for the research. They provide a lot of development data, and give their time to answer our questions. We thank all these engineers.

REFERENCES

Anton, A. I. (1996). Goal-based requirements analysis. In *Proc. of the Second International Conference on Requirements Engineering (ICRE'96)*, pages 136–144. IEEE.

Aoyama, K., Ugai, T., Yamada, S., and Obata, A. (2007). Extraction of viewpoints for eliciting customer's requirements based on analysis of specification change records. *APSEC*, pages 33–40.

Arkley, P. and Riddle, S. (2006). Tailoring traceability information to business needs. In *Proc. of the 14th International Requirements Engineering Conference (RE'06)*, pages 239–244. IEEE.

Beck, K. and et al. (2001). Manifesto for agile software development, (url http://agilemanifesto.org/).

Dardenne, A., van Lamsweerde, A., and Fickas, S. (1993). Goal-directed requirements acquisition. *Science of Computer Programming*, 20:3–50.

Fricker, S., Gorschek, T., and Myllyperkiö, P. (2007). Handshaking between software projects and stakeholders using implementation proposals. *Requirements Engineering: Foundation for Software Quality*, 4542:144–159.

Houdek, F. and Pohl, K. (2000). Analyzing requirements engineering processes: A case study. In *Proc. of the 11th International Workshop on Database and Expert Systems Applications (DEXA'00)*, pages 983–987. IEEE.

Jacobson, I., Booch, G., and Rumbaugh, J. (1999). *The Unified Software Development Process*. Addison-Wesley.

Kousik, S. R. and Raman, V. (2007). Total requirements control at every stage of product development. In *Proc. of the 15th International Requirements Engineering Conference*, pages 337–342. IEEE.

Lehman, M. M. (1978). Laws of program evolution -rules and tools for programming management-. In *Proc. of the Infotech State of the Art Conf., Why Software Projects Fail?*, pages 11/1–11/25. Program Press.

Lehman, M. M. (1996). Feedback in the software evolution process. *Information and Software Technology*, 38(11):681–686.

Nakamura, T. (2005). Analysis of project management reports of 49 system integration projects. In *Proc. of the 13th International Requirements Engineering Conference (RE'05)*, pages 485–486. IEEE.

Nakatani, T., Hori, S., Ubayashi, N., Katamine, K., and Hashimoto, M. (2008). A case study: Requirements elicitation processes throughout a project. In *Proc. of the 16th International Requirements Engineering Conference (RE'08)*, pages 241–246. IEEE.

Robertson, S. and Robertson, J. (1999). *Mastering the Requirements Process*. Addison-Wesley.

Sommerville, I. (2005). Integrated requirements engineering: A tutorial. In *IEEE Software*, pages 16–23. IEEE.