Title	Requirements engineering in knowledge intensive manufacturing shop floors: a knowledge-based approach
Author(s)	Hoang, Le Anh
Citation	
Issue Date	2020-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/16371
Rights	
Description	Supervisor: , ,

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

Master's Thesis

Requirements engineering in knowledge intensive manufacturing shop floors: a knowledge-based approach

1730019 HOANG LE ANH

Supervisor: Prof. Naoshi Uchihira
Main examiner: Prof. Naoshi Uchihira

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
Master of Science (Knowledge Science)

February 2020

Acknowledgement

Hoang Le Anh
Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology (JAIST)
February 2020

Abstract

Today's manufacturing industry faces hard competition. To minimize the fluctuations in demand or economic downturns and to reduce labor cost, companies resort to low-cost outsourcing or hiring contract operators. Manufacturing shop floors are always knowledge-intensive environment where highly skilled workforce is led by meisters (experienced and masterful operators or line leaders) with their strong tacit knowledge. Outsourcing provides the benefit of gain in business profitability with trade-off of skill transfer and knowledge inheritance due to high turn-over rate. With the emergence of Industry 4.0 and smart factory concept, externalizing this tacit knowledge and automating tasks using IT systems is a promising solution. Requirements engineering is an essential part of software development. Explicit requirements are usually clearly described and well aware-of by target users. Tacit requirements are hidden or embedded which cover the skills and experience of shop floors' meisters. To complete project in time, on budget, it is essential to fully capture and implement these tacit requirements. This thesis constructs a framework design which integrated ethnography, knowledge externalization process with the tacit requirements elicitation to achieve this goal. A case study of a semiconductor factory is used to illustrate and evaluate this approach.

Keywords: knowledge conversion, Requirements Engineering, ethnography, project management; tacit knowledge

Table of contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research background

Today's manufacturing industry faces hard competition. To minimize the fluctuations in demand or economic downturns and to reduce labor cost, companies resort to low-cost outsourcing or hiring contract operators. Manufacturing shop floors are always knowledge-intensive environment where highly skilled workforce is led by meisters (experienced and masterful operators or line leaders) with their strong tacit knowledge. For example, experienced quality assurance engineers can detect micron level scratches with naked eyes or measure the curvatures of a metal surface just by touching. In other case, a skillful operator can make fine adjustments on equipment with hundreds of parameters just according to his feel to smooth the production process. Outsourcing provides the benefit of gain in business profitability with trade-off of skill transfer and knowledge inheritance due to high turn-over rate. With the emergence of Industry 4.0 and smart factory concept (Kagermann, 2015), externalizing this tacit knowledge and automating tasks using IT systems is a promising solution. All system developments begin with requirements elicitation. In this phase requirement engineers interact with users, customers and other stakeholders to discover basic business process, gather related domain knowledge and then structure those requirements in a way that can be used for the design specifications and for further classification, prioritization as well as verification in future iterations. This process is referred to as requirements elicitation (Sommerville, 2011). Figure 1 shows how requirements elicitation is related to the whole requirements engineering process. Requirements typically are divided into two categories: explicit and tacit. Explicit requirements are usually clearly described and well-aware of by target users. Contrastingly, tacit requirements are often hidden, or implied that these are automatically included in without any clear statement (Onyeka, 2013). In the context of manufacturing shop floors, tacit requirements cover the skills and experience of shop floors' meisters mentioned above. To complete project in time, on budget, it is essential to fully capture and implement these tacit requirements (Boehm, 1984).
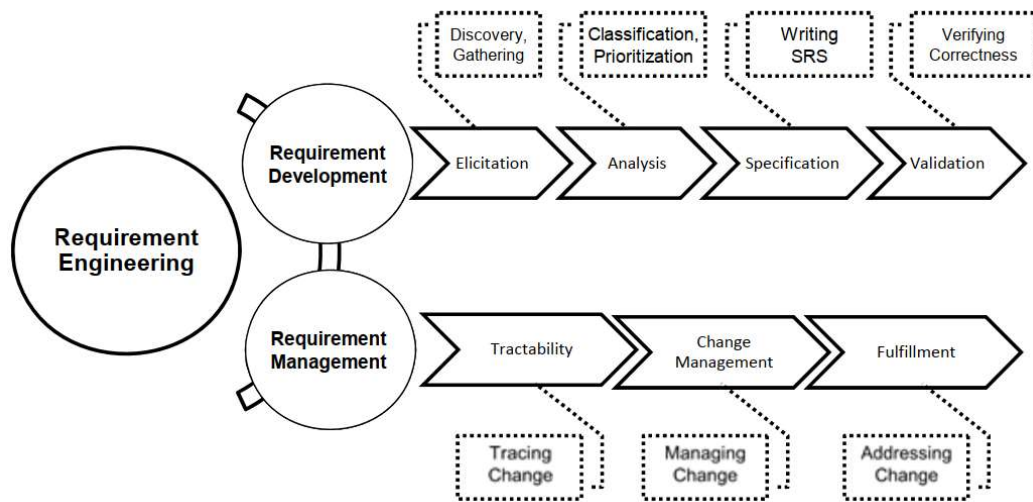
Figure 1 Requirements Engineering's scope (Sommerville, 2011)

1.2 Requirements engineering at manufacturing shop floors

The first and foremost tasks of requirements engineering method is requirement elicitation. There are numerous ways to perform this task. Commonly used methods are interviews, questionnaires, surveys, or the analysis of existing documentation. The goal is to have a comprehensive understanding of what stakeholders do and how they may use the system. During requirement elicitation, the requirement engineer works with stakeholders to understand the application domain, discover all concerning work activities, services and system functions that stakeholders wish to implement as well as the required performance of the system and hardware constraints, etc. (Sommerville, 2011). In manufacturing shop floors' automation, manufacturing systems play a central role. These systems coordinate production operations with centralized management of production equipment, processing recipes, machining conditions and optimizing production processes by advanced scheduling, automated work instruction as well as integration with material planning, demand forecast and inventory data. These systems also offer traceability by capturing the beginning and completion of every work instruction. In contrast, an enterprise system is a cross- functional information system that coordinates and integrates all key business processes and functions of the whole organization. The target functions are wide ranged, covering finance, accounting, human resources, transportation, distribution and customer relationship management such as quotation, purchasing, billing, planning the resources of an organization etc. However, due to the specialized nature and product/process- dependent characteristics of manufacturing systems, the complexity and the level of difficulty of Requirements Engineering are different for each type of the system as shown in Table 1.

10

Manufacturing systems have diversified stakeholders with many different backgrounds, interests and expectations. This further complicated the tacit requirements elicitation process.

Table 1 Requirements engineering: manufacturing systems vs. business systems

	Manufacturing system	**Business system**
Main stakeholders/ Users	Production Management Dept, Quality Management Dept, Machine maintenance Dept, Shop floor's operators	Accounting Dept, Procurement Dept, Human resources Dept, Logistics Dept, Sales Dept, Production Management Dept
Domain knowledge	Manufacturing process (specialized and process/product dependent)	Business process (standardized, general functions and procedures)
Expected level of automation	Moderate (in some cases, some human intervention is introduced to optimize the result)	Moderately high (with recent progress in RPA and AI technology, some standardized tasks can be fully automated)
Tacit requirement	Large amount Highly concentrated on shop floor and possessed by experienced and skillful operators	Low amount due to standardized process
Requirement elicitation	Complicated due to specialized domain knowledge and tacit requirement	Less complicated since standardized processes can be defined with predefined templates

To summarize, Figure 2 general constraints which software developers often encounter when building system for manufacturing shop floors. First, since end users are the manufacturing shop floors' operators, line leaders, production managers etc., the user's environment is inherently knowledge-intensive especially in terms of tacit knowledge. Secondly, depending on the products which the target shop floor is producing, knowledge domain varies widely. In some case the domain knowledge is so specialized (such as semiconductor, flat panel display etc.) that it takes great efforts for software engineer to understand the process. Thirdly, target systems are manufacturing systems which are complicated and product dependent. Finally, in industry environment, on-site system engineers have limited access to manufacturing shop floors or no contacts with shop floors' users at all due information security policy of the company.
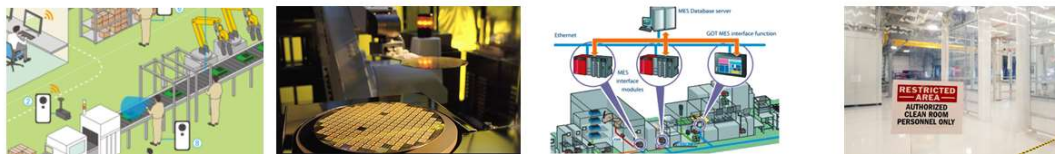
Figure 2 General constraints of software system development in manufacturing

1.3 Research objectives and research questions

As shown in section 1.2, there are myriad of challenges in Requirements Engineering at manufacturing shop floors. Software engineering has a long history with a huge stock of established methodologies not only in the whole software development process (Agile, Waterfall etc.(Figure 3)) as whole but also in Requirements Engineering. Yet, project failures still often occurred and a large part of that is due to requirement acquisition failures (ranked in top 3 causes (Figure 4)).
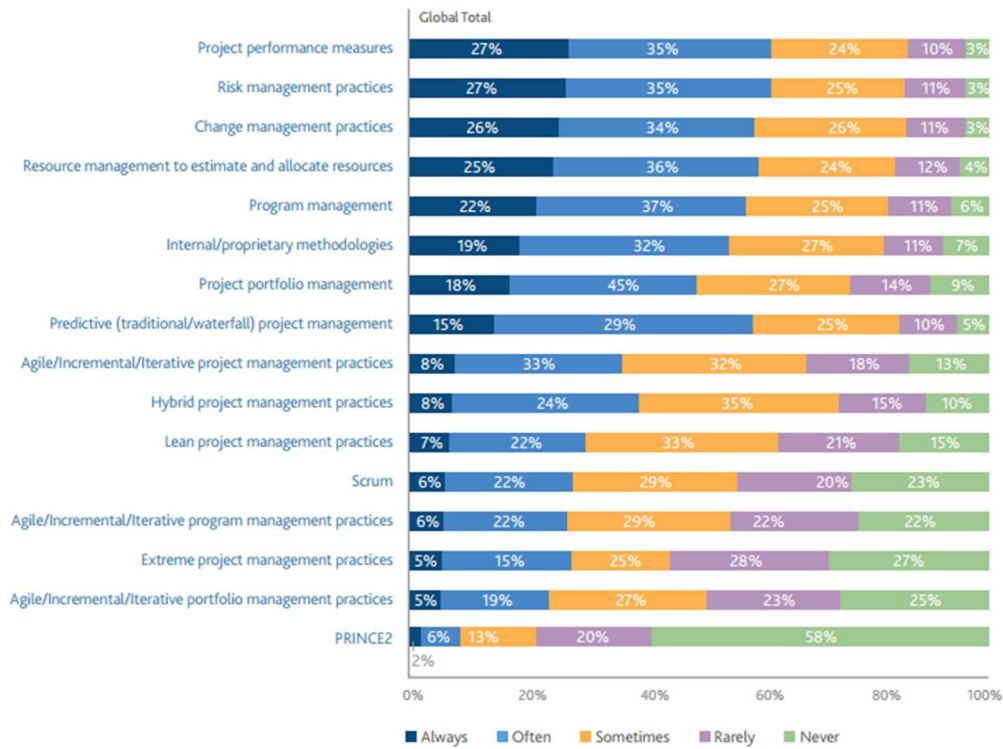
Figure 3 Project Management Practices and Usage(Project Management Institute, 2018)
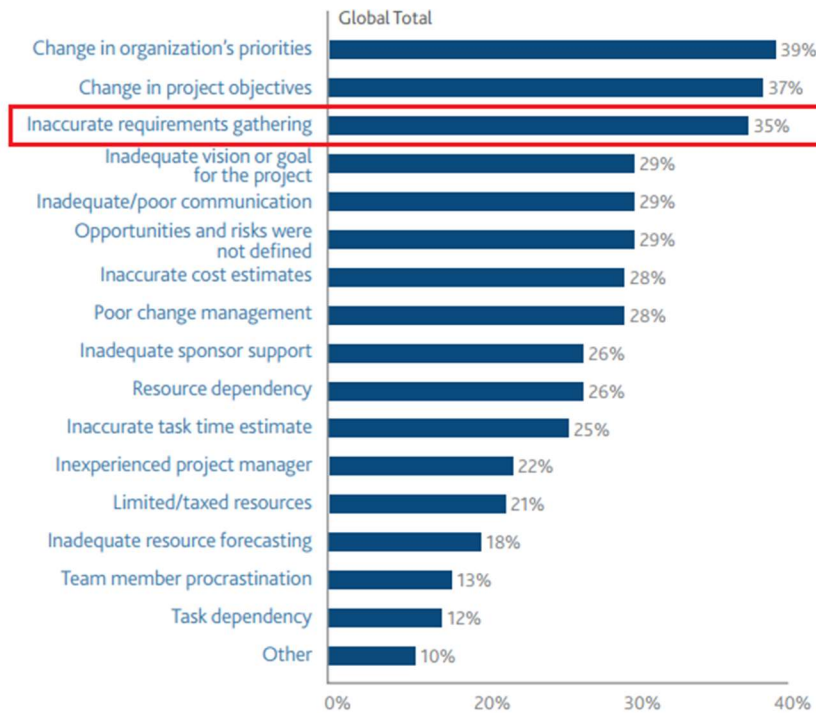
Figure 4 Primary causes of project failures(Project Management Institute, 2018)

"What is the reason that fully financially supported and experienced software development teams fail to deliver systems which meet requirements, on time, within budget?", "What can we do to improve this?" have been intriguing questions that leads to this research. The purpose of this research is to investigate and construct a framework design which integrates ethnography and the knowledge creation process (especially the process of conversion from tacit knowledge to formal knowledge) to fully capture and implement tacit requirements at manufacturing shop floors. Furthermore, this research examines the issues and effects of applying the proposed framework to the acquisition of tacit requirements at manufacturing shop floors. We verified the framework by performing a case study analysis on the development of a scheduling/dispatching system at a semiconductor factory in Japan.

With that in mind, this thesis attempts to answer the following research questions:

Main research question (MRQ):

In knowledge-intensive manufacturing shop floors, what characterize an effective knowledge conversion model for tacit requirements elicitation?

Subsidiary research questions (SRQ):

SRQ1: What are the challenges of acquiring tacit requirements at the manufacturing shop floors?

SRQ2: What role does the knowledge creation process (especially the tacit-to-explicit knowledge conversion process) play in acquiring tacit requirements?

SRQ3: What is an effective process for capturing implicit requirements in knowledge-intensive manufacturing shop floors?

1.4 Research methodology

This study adopts general guidelines on research provided by Kothari (2004). To address the subsidiary research question "SRQ1: What are the challenges of acquiring tacit requirements at the manufacturing shop floors?", a literature review of previous research on tacit requirement acquisition is conducted and then a participant observation of software development process at the case study sites is performed. Similarly, "SRQ2: What role does the knowledge creation process (especially the tacit-to-explicit knowledge conversion process) play in acquiring tacit requirements?" are explored by surveying previous works on the application of knowledge creation process, knowledge conversion cycle in Requirements Engineering in general as well as participant observation of software development process in views of knowledge management. Finally, a comprehensive analysis is performed to point out limitations of previous

research. Based on the analysis results, we proposed a new method for tacit requirements acquisition requests and conduct verification in the actual system development process. Verification and evaluation of the method is performed as a case study at a semiconductor factory in Japan. Results are collected in terms of actual man-hours and hearing of issues at development sites and manufacturing shop floors.

1.5 Organization of the thesis

This thesis is organized as follows:

Chapter 1 briefly introduce the research background, problem settings, research questions and the organization of this thesis.

Chapter 2 elaborates on past research which explains fundamental concepts and theoretical background of knowledge creation theory, Requirements Engineering and ethnography which forms the foundation of this thesis' theoretical framework.

Chapter 3 describes the newly proposed dual loop knowledge conversion model for requirements engineering at knowledge-intensive manufacturing shop floors, which is the main theoretical contribution of this thesis.

Chapter 4 explains details of the case study in which the author performed the verification and evaluation of the newly proposed knowledge conversion model

Finally, Chapter 5 concludes the thesis by summarizing answers to research questions. This chapter ends with a discussion on the remaining issues due to the limited scope of this research and potential future works.

Chapter 2

Literature Review

This chapter explains fundamental concepts and theoretical background of knowledge creation theory, Requirements Engineering and ethnography which forms the foundation of this thesis' theoretical framework.

2.1 Requirements Engineering (RE)

Software requirements

Software requirements are the predecessor of all other phases of software development, including software design, implementation, and testing. Institute of Electrical and Electronics Engineers standard 610.12-1990 ''Glossary of Software Engineering Terminology'' provides the most primitive form of definitions of software requirements as following:

Requirement:

(1) A condition or capability needed by a user to solve a problem or achieve an objective.

(2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.

(3) A documented representation of a condition or capability as in (1) or (2)

(IEEE 610.12, 1990)

In the above-mentioned primitive form, definitions suffer from the certain limitations. First, only needs – that is, only the essential aspects – are mentioned. Users expect the developers to bring expertise to deliver the software product are not taken into consideration. Secondly, there is no regard to constraints of the users. Thirdly, Interfaces with other/existing systems are not mentioned. Fourth, responsibility for the requirements is not clearly specified. This definition does not include users who can state needs. Therefore, the developers are easily misled and missed out on some of the stakeholders involved. Finally, *unstated needs* (requirements which are not documented) will not be considered. Based on the above discussion, in the context of software development, we can define software requirements in a more comprehensive manner for this thesis:

Software requirements describe all needs, expectations, features or qualities at user level, general system/product properties, system/product specific constraints or interfaces which must be delivered by the software.

Requirements Engineering Concept

Requirements Engineering is the earliest phase of the software development life cycle. Zave (1997) outlines requirements engineering as the branch of software engineering addressing the real-world goals (such as functions and constraints) of software system. Pohl (1996) pointed out during the software development process, requirements engineering helps eliciting the users' requirements, documenting the requirements in a comprehensible format, validate, verify and manage requirements through entire life cycle of the system. More generally, requirements engineering does not address only technical issues, but also organizational and managerial issues. It is an essential step to design software that meets the expectations of users. This is executed by clarifying the needs of stakeholders, understanding the context, modeling, negotiating, documenting, validating requirements to make sure those match the negotiated requirements, and managing requirements change. Stakeholders here refer to end users, customers, decision makers, external systems and developers who are involved in the software development. Usually, stakeholders possess different background knowledge, and have different organizational goals and views due to different relative positions in the organization. Therefore, it is impossible to produce a complete, consistent and well-structured set of requirements if there are conflicting sources of information, discrepancies in understandings of needs and expectations until consensus in reached between stakeholders. In summary, we cannot complete the requirements specification without considering the physical and organizational environment where the software system will be installed and used

Requirements Engineering Process (REP)

Figure 5 shows the outline of a typical Requirements Engineering Process defined by Sommerville (2011). According to this, there are three main activities in the requirements engineering process:

Requirements elicitation and analysis: Deriving the system requirements through observation of existing systems, discussions with potential users and procurers, task analysis, etc. This may involve the development of one or more system models and prototypes.

Requirements specification: Translating the information gathered during requirements

elicitation analysis into documents. Different techniques are used to document the requirements by using natural language or conceptual models.

Requirements validation: Checking the documented requirements for accuracy, consistency, and completeness. This activity also involves some negotiation with stakeholders to refine the requirements.
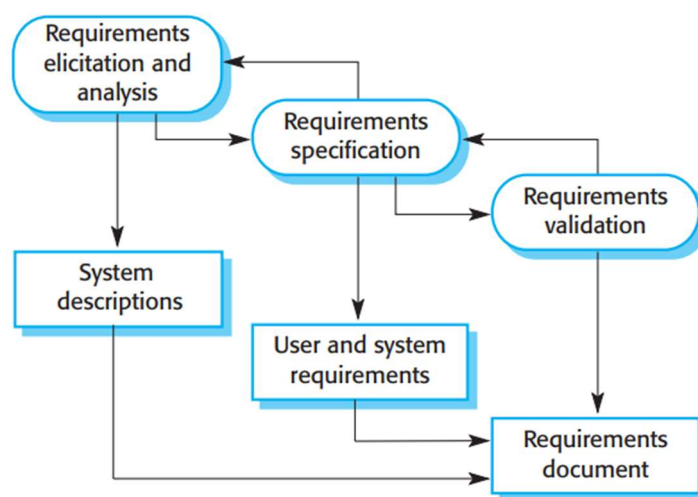
Figure 5 Requirements Engineering Process(Sommerville, 2011)

Although this definition does not include another important aspect of Requirements Engineering which is Requirements Management.

RE in the software development life cycle

Regardless of Software Development Life Cycle (SDLC) model - be it pure Waterfall/V-model (Forsberg and Mooz, 1991; Royce, 1987), Incremental, Agile methods like eXtreme Programming (Pressman, 2005; Sommerville, 2011), Scrum (Sutherland, 2004), Kanban(Anderson, 2010; Ladas, 2008) or some hybrid – RE activities are the same. The only difference is that with each SDLC model, the timing RE activities are performed and the degrees of depth or detail of these activities. As depicted by Figure 6, Waterfall/V-model is a sequential software development process in which main phases of requirement definition/modeling, design, implementation and testing, integration and testing, and maintenance are performed step-by-step running downwards like a waterfall through. As shown in Figure 7 and Figure 8 or Figure 9 agile methods like Incremental, extreme programming or scrum repeat the traditional phases of Water fall model in fast and small cycles while encouraging building prototypes, interaction between managers, developers and users to better refine the end product and make modifications as well as adjustment of requirements during each cycle.

18
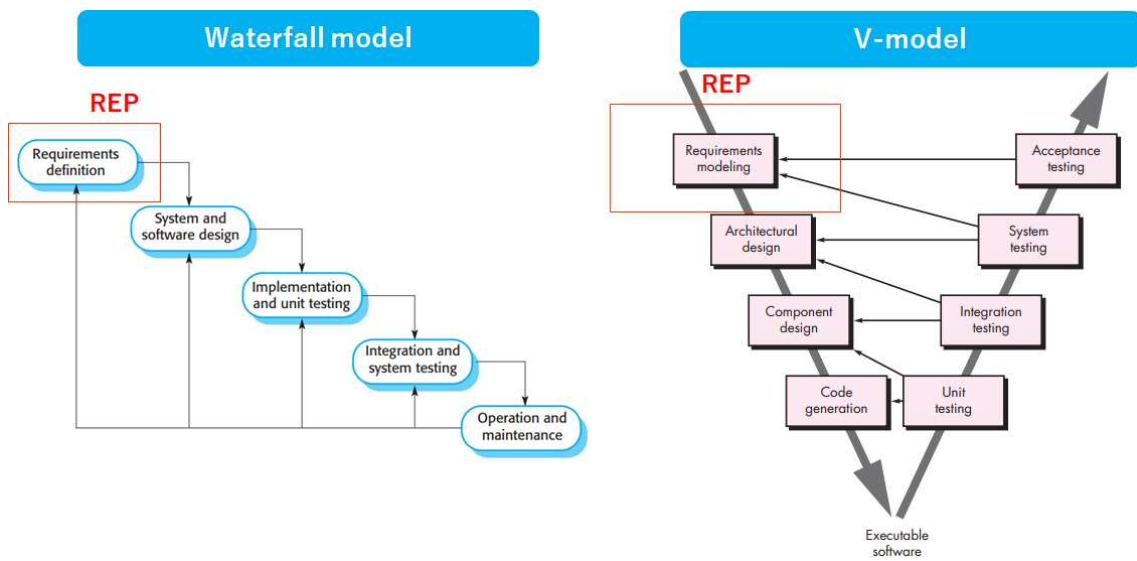
Figure 6 REP's position in Waterfall and V-model SDLC model (Forsberg and Mooz, 1991; Royce, 1987)
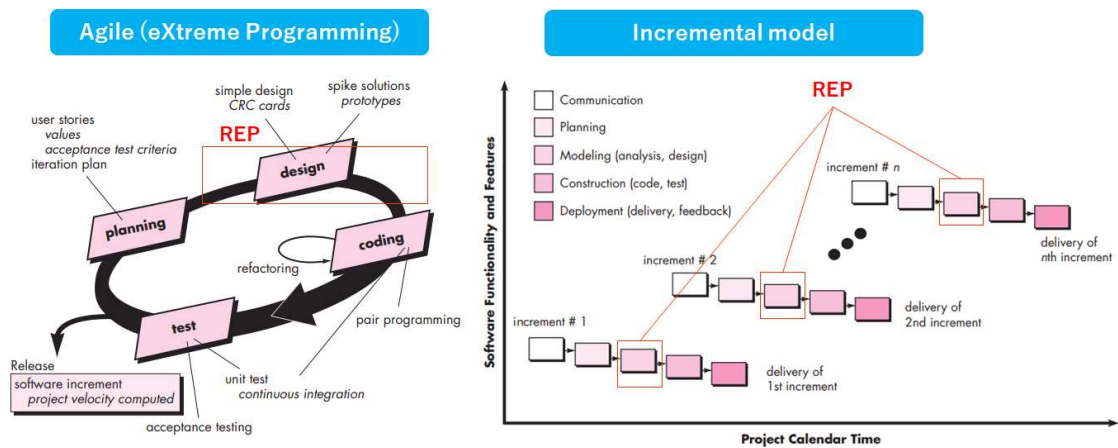
Figure 7 REP's position in eXtreme Programming and Incremental SDLC model (Pressman, 2005; Sommerville, 2011)
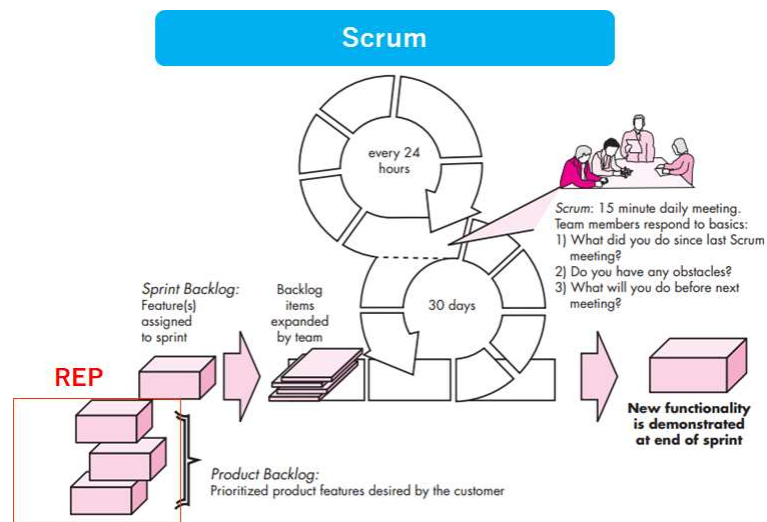
Figure 8 REP's position in Scrum SDLC model (Sutherland, 2004)
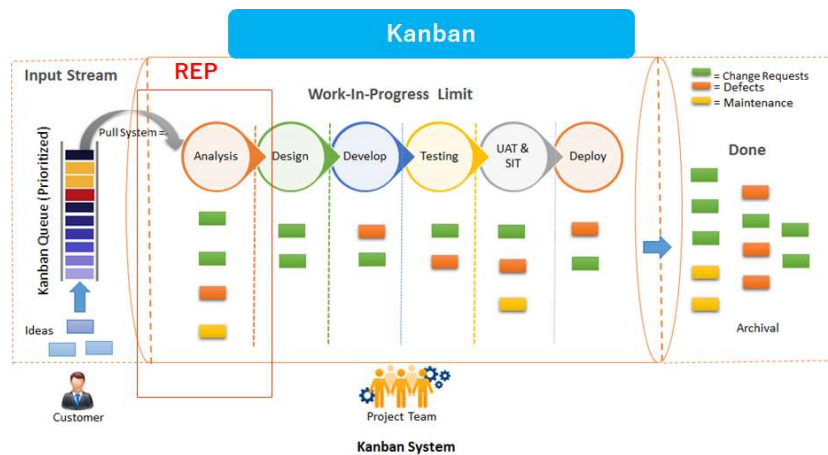
Figure 9 REP's position in Kanban SDLC model (Anderson, 2010; Ladas, 2008)

The importance of REP

As shown in Chapter 1, projects failed to satisfy the stakeholder largely because of poor requirements. Frederick P. Brooks (1987) emphasized that the most difficult part of creating software is deciding correctly what to build. This is a difficult conceptual work, especially, devising detailed technical requirements which include every interface to users, machines, and to external systems. If this part is done wrong, the resulting system will be damaged seriously and fixing this will take an enormous cost. In view of such difficulties, Requirements engineering process is a crucial part of the whole software development cycle.

2.2 Knowledge management

The concept of knowledge

In general, knowledge is an abstract concept. For any person, there are always things they claim to know, and things they do not. But what is more important is there are some things the person really knows and some things they do not. "What does it take to know something?" has long been a source of debate in philosophy. When we say we know something, it is not enough just to believe it—we do not know things which we are wrong about. Knowledge seems to be more concerned with how we get at the truth of everything. The most basic definition of knowledge evolves around this concept. One of the most commonly used definition of knowledge is "justified true belief" proposed by Nonaka and Takeuchi (1995, p. 87). According to this definition, knowledge entails three basic conditions which are called the tripartite account of knowledge. According to Neta and Pritchard (2009)These conditions are the following :

- The truth condition: Someone knows a proposition if and only if that proposition is true.
- The belief condition: If someone knows a proposition then he/she must believe that proposition.
- The justification condition: There must exists a method to justify that the belief is true.

However, this is still too abstract to use with regards to the research target of this thesis which is software engineering for the manufacturing shop floor. In the context of this thesis, we focused more on literature about organizational knowledge management. We employed the definition proposed by Alavi and Leidner (2001): knowledge represents personalized information possessed in the mind (not necessarily original, unique, useful, or accurate) related to ideas, observations, judgments, concepts and procedures.

Tacit knowledge and explicit knowledge

Polanyi (1967) advocated that human knowledge starts from the fact that we can know more than we can tell. This literature for the first time in history highlights an important problem in knowledge management. That is, as human sometimes we are unable to spontaneously articulate knowledge that we take for granted. Hence, we have the classification of knowledge into two types: tacit knowledge and explicit knowledge (Table 2). **Explicit knowledge** refers to the artificial knowledge which is easy to express explicitly with formal and systematic language, transfer and share with others

using manuals, words in the paper and compiled data (Nonaka and Takeuchi, 1995; Nonaka, Toyama, and Nagata, 2000). Recent research pointed out this also includes scientific formulae (Seidler-de Alwis and Hartmann, 2008), instructions, facts and figures (Mahr and Lievens, 2012) or specifications for product manufacturing (Loebbecke, van Fenema, and Powell, 2016). In contrast, **tacit knowledge** is difficult to articulate (Nonaka and Konno, 1998; Nonaka and Takeuchi, 1995; Nonaka et al., 2000). This type of knowledge is embedded in individuals (Argote and Ingram, 2000) and associated with personal experience (Loebbecke et al., 2016). It represents skills and "know-how"'s which are formed by perspectives, beliefs, insights and intuition, sometimes feelings and thinking (Popadiuk and Choo, 2006). Tacit knowledge facilitates individuals in the organization to effectively communicate, cooperate to achieve common goals of the organization (Argote and Ingram, 2000; Saint-Onge, 1996).

Table 2 Characteristics of tacit knowledge and explicit knowledge

Tacit knowledge	Explicit knowledge
• *Ideas, beliefs, values, attitudes, perspectives* (Nonaka and Konno, 1998; Nonaka and Takeuchi, 1995) • *Company culture, social norms, organizational politics* (Joe, Yoong, and Patel, 2013) • *Skills and know-how* (Nonaka and Konno, 1998; Nonaka and Takeuchi, 1995) • *Experience* (Loebbecke et al., 2016) • *Contextual thinking and feeling* (Popadiuk and Choo, 2006)	• *Software, manuals or specifications* (Nonaka and Konno, 1998; Nonaka and Takeuchi, 1995) • *Formulae, equations, recipes* (Seidler-de Alwis and Hartmann, 2008) • *Facts and figures, guidance and instructions* (Mahr and Lievens, 2012) • *Technical specifications for production etc.* (Loebbecke et al., 2016)

2.3 The knowledge creation theory

Basically, knowledge creation is the act of developing new content and replacing the existing content of organizational knowledge. Knowledge creation theory is first proposed by Nonaka and Takeuchi (1995). The authors based on comprehensive research on Japanese companies, defined the knowledge conversion process as the SECI model, which achieved paradigmatic status since its publication. SECI has four conversion modes: Socialization, Externalization, Combination and Internalization (Figure 10). These four modes help visualize the model of knowledge flow and represents it as knowledge flow in dimension of explicitness. Socialization is the

sharing of tacit knowledge and experience between individuals through social interaction such as apprenticeship and on-job training. Externalization is a process which articulates and makes clear knowledge from tacit knowledge to new explicit concepts such lessons learned, documented procedures. In the externalization mode, knowledge collected by individuals is transformed into collective mindsets. In the combination mode, we create new explicit knowledge by synthesizing or reconfiguring the existing explicit knowledge. The new explicit knowledge should be more complex and systematic than the existing knowledge – this is the result of knowledge creation. Like traditional learning - learning as a result of reading, internalization mode is illustrated by the creation of tacit knowledge from explicit knowledge. Internalization mode is the knowledge transformation process in which knowledge is broken down from collective level (belonging to the group/the mass) to individual level. The SECI model has been widely accepted in the knowledge management field and used in many domains other than knowledge management such as human resources and leadership (Bell DeTienne, Dyer, Hoopes, and Harris, 2004), generation of new product ideas (Schulze and Hoegl, 2008), and social networks (Akiyoshi, 2008; Ismail and Ahmad, 2011; Phosaard and Wiriyapinit, 2011). Furthermore, it is crucial to bear in mind that tacit knowledge and explicit knowledge are not entirely two different things. Nonaka *et al.* (2014) argued that tacit knowledge is the basis of all knowledge because even the most explicit knowledge still contains some tacit parts or aspects. This thesis reframes tacit-to-explicit knowledge conversion to discuss how software requirements are formulated and articulated from users/stakeholders of the system.
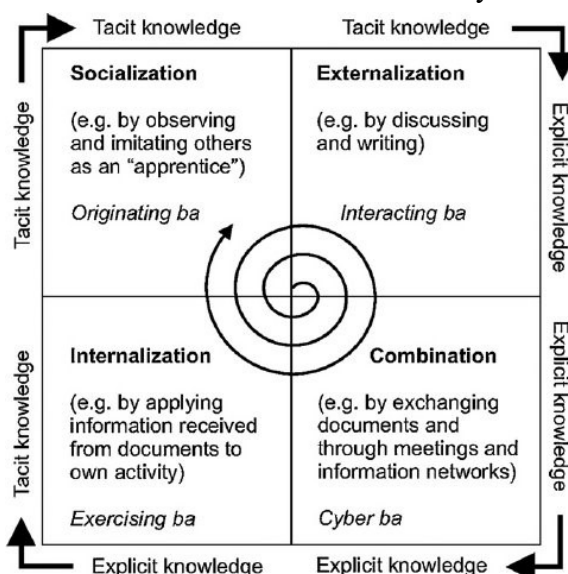
Figure 10 SECI Model(Nonaka and Takeuchi, 1995)

2.4 Tacit requirements

Tacit aspects of software requirements

The roots of unmentioned requirements are the unstated needs or take-for-granted assumptions. Missing these gaps until much later phases can have devastating results for projects. As mentioned in section 1.2, in the context of manufacturing shop floor, missed gaps are often related to the following types of knowledge:

- **Tacit knowledge**: Line operators answered "I forgot to mention our electrical power auto adjusting system. I'm so familiar to the fact it is always running in the background as a routine. Therefore, I do not even notice it or have it in my mind anymore".

- **False assumptions**: Line operators answered "Oops, by saying "the system take into consideration the beam current, to extend the ion source life, and to shorten the setup time", we assumed that the system will automatically change the threshold value to assign lots when the product type changed.

- **Missing manuals/Documented knowledge that is not transferred properly:** Contracted operators answered: "I just joined the company 2 weeks ago. I just followed what seniors are doing. Nobody knows why the system works that way. The person who made these rules doesn't work here anymore, but he might have documented this and left it somewhere before he left". (Intended meaning: For further miniaturization of the IC, it is important to control ion implantation precisely. The control is realized by a variety of factors such as dose uniformity and implant angle, elimination of device damage caused by the charge-up during implantation, filter-out of various types of contamination, generation of new ion beams, and patterning implantation to compensate the variation of transistor characteristics across the wafer. That is why we charge product types in order of A⇒B⇒C).

Consider the situation in which someone who took piano lessons many years ago, tries to perform a difficult classical piece again. This can only be ascertained by trying to play it again. First, he or she let the movements of fingers go with the flow over the keyboard, rerun the habitual sequence of movements forms a kinesthetic, proprioceptive pattern which he or she absorbed into the mind the body years before. Such kind of

experiences suggested that "the body" knows something that a wide-awake mind may not be able to recall. This supported the assertion that the body as memory of practice and affirm the role the tacit knowledge. At manufacturing shop floors, we also witness the same thing. When handling a machine, experienced operators run a non-stop habitual sequence of button pushing, knob adjusting for parameter optimization before material processing starts skillfully. But there are no documentations of standard procedures, and the experience operators do not even notice and explain it verbally when interviewed because he is so familiar to the habits or routines. The common characteristics of tacit requirements can be summarized as shown in Table 3. On linguistic level, tacit requirements are difficult to express and communicate. Regarding the domain knowledge, tacit requirements are specialized and application domain dependent. Tacit requirements often stem from users' tacit knowledge (especially in the context of manufacturing shop floors). Tacit requirements are based on knowledge and experience which concerned stakeholders accumulates in practice during a long period of time (Basir and Salam, 2015). Comparing the above characteristics with that of tacit knowledge gives us one-to-one matching similarities as shown in Table 3. Therefore, tacit requirements elicitation represents a type of tacit to explicit knowledge conversion.

Table 3 Tacit requirements vs. tacit knowledge

Tacit requirements	Tacit Knowledge
Tacit requirements are difficult to articulate express, communicate and share	Tacit knowledge is personal, non-linguistic, non-numerical (Nonaka and Takeuchi, 1995)
Tacit requirements are application domain dependent.	Tacit knowledge is context-specific and deeply rooted in personal experiences, ideas, values and emotions (Nonaka and Takeuchi, 1995)
Tacit requirements usually stem from users' tacit knowledge	Tacit knowledge covers tacit requirements
Tacit requirements are based on knowledge and experience which concerned stakeholders accumulates in practice for a long time(Basir and Salam, 2015)	Tacit knowledge is subjective and personal knowledge (Janz and Prasarnphanich, 2005)

Ambiguity of software requirements

During the first phase of elicitation, requirements are mostly communicated verbally by natural language. It is easier to correctly transmit information from stakeholders to developers or between stakeholders if they all speak the same language, have the same background, experiences, area of expertise. However, in real life such ideal situation

hardly exists. Human perception makes natural transformations of information that is language-based ambiguous by simplifying the expression of knowledge in colloquial conversation, descriptions of related phenomena which assumes implicit background knowledge or tacit knowledge. Consider the following example which was given by Parnas, Asmis, and Madey (1991) when constructing software for a nuclear power plant. In this example, the requirement was introduced without the requirement engineers having been told that the water level varies continuously. A "close the pumps if the water level remains above 100 meters for more than 4 seconds" requirement implied at least four interpretations (1) the upper limit of the *mean water level* over the 4 second period is 100 meters, (2) the upper limit of the *median water level* over the 4 second period is 100 meters, (3) the upper limit of *the root mean square* water level over the 4 second period is 100 meters (4) *the minimum water level* over the 4 second period is 100 meters. The software engineers did not pay attention to this ambiguity and quietly assumed the fourth interpretation. However, under this interpretation, waves in the tank will make the water level to overshoot the maximum level without triggering the shut off function. This is a clear demonstration of how the interpretation of the ambiguity depends on readers' background. Because there are no documentations mentioning waves and software engineers have no experiences or knowledge on waves in the tank there is no way to detect this omission. Berry and Kamsties (2004) based on research in linguistics and software engineering classified requirements ambiguity as shown in Figure 11.
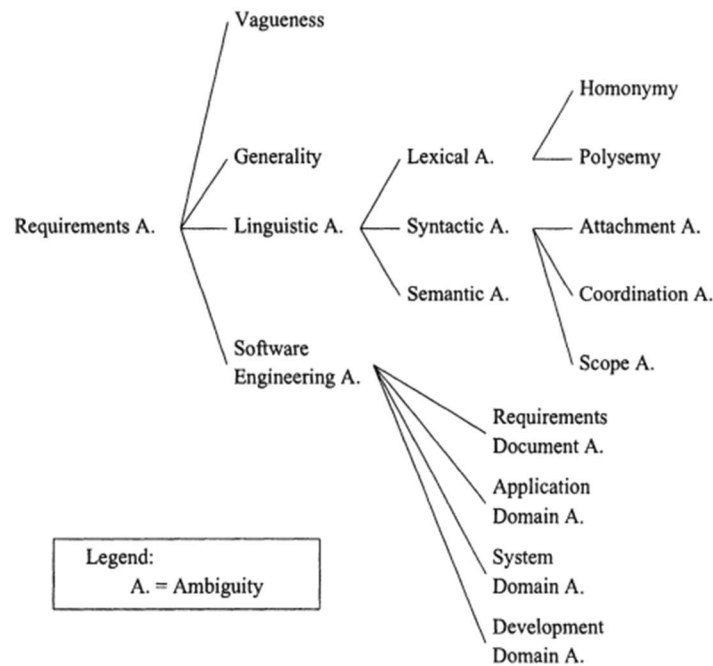
Figure 11 Taxonomy of Ambiguity Types .(Berry and Kamsties, 2004)

Suggestions to minimize ambiguity during requirements elicitation are two strategies (1) establishing a context and make explicit and agreed to by all the stakeholders (2) the requirements engineer's paraphrase what she understood from the customers' and users' statements in her own words is effective way for the requirements engineer to get the customers and users to spot their own ambiguities. Several communication techniques have been proposed by Bostrom (1989) to support this. In summary, requirements ambiguity further complicates the process of requirement elicitation making tacit requirements capture even more difficult. There is an urgent need for devising an effective tacit requirements elicitation method to address this issue.

Challenges in tacit requirements elicitation

With all the characteristics analyzed in section 2.5.1 and the complication due to ambiguity in natural language as specified in section 2.5.2, tacit requirements usually pose a big challenge in elicitation process. Figure 12 demonstrates a common scenario with an example of Requirements Engineering for manufacturing shop floors. Stakeholders (or system's users) are production managers, line leaders and machine keepers. There are four issues which hinder the process of tacit requirement elicitation: (1) tacit requirement that is not made explicit by stakeholders/users but presumably implied in requirements by users, (2) tacit requirement that is never mentioned by the stakeholders/users, (3) inconsistent interpretation of requirements due to ambiguous

nature of tacit knowledge possessed by different classes of users,(4) On-site system engineers (who are outsourced vendors) have limited access to manufacturing shop floors due to data security, protection and confidentiality policy of the company, therefore useful participant observations are near impossible.
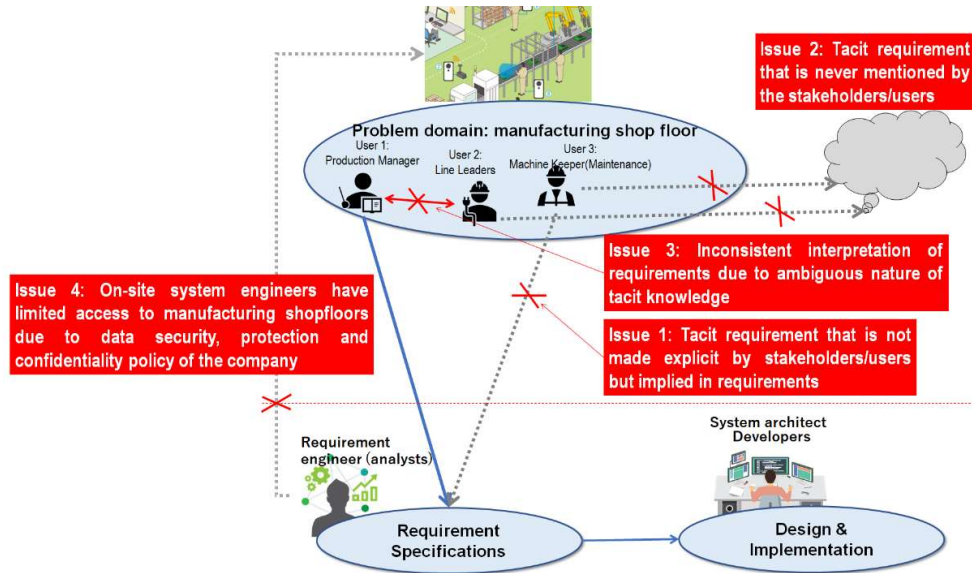
Figure 12 Challenges in tacit requirements elicitation at manufacturing shop floors

2.5 Knowledge-based approach for REP

Reframing REP in view of knowledge creation

Von Krogh, Ichijo, and Nonaka (2000) argued that there are five knowledge creation enablers: (1) Instill a knowledge vision, (2) Manage conversations, (3) Mobilize knowledge activists, (4) Create the right context, and (5) Globalize local knowledge. The effective knowledge creation depends on an enable context or shared space that fosters emerging relationships (because knowledge is dynamic, relational, and rather than on absolute truth or hard facts). This fits well with REP as shown in Table 4.

Table 4 REP's knowledge creation enablers

Enabling factors (Von Krogh et al., 2000)	REP's equivalence (in the context of manufacturing shop floors)
(1) Instill a knowledge vision	REP's objectives: (1) Modeling real world using systems, (2) Create final requirements specification (SRS) as a storage of knowledge
(2) Manage conversations	Organize discussion and negotiation between users and developers
(3) Mobilize knowledge activists	Requirement engineers gather with stakeholders

(4) Create the right context	Focus on the user environment (manufacturing shop floors)
(5) Globalize local knowledge	Design standardized features so the systems can be reuse

Does requirement engineering process create new knowledge? The answer is "Yes". Let's reframe the requirement engineering process as shown in Figure 13 (Kotonya and Sommerville, 1998). This illustrates the requirements engineering process flow with main activities and each activity's inputs. Stakeholders' viewpoints are knowledge sources. Different user classes or roles in the organization have different ways to express their knowledge about the problem domain. Their understanding of situational context also varies through the whole elicitation process. Adding complexity to this issue is the diversity of the requirements analysts' experience. Requirements elicitation normally is conducted in a set of sessions, in parallel or in sequence. New understanding about the desired system is extracted, recorded and analyzed and combined with understood knowledge to produce "new understood knowledge". This new understood knowledge will finally be reflected in requirements analysis and final requirements specification (SRS). The above analysis shows that the Requirements Engineering process is some form of knowledge creation process.
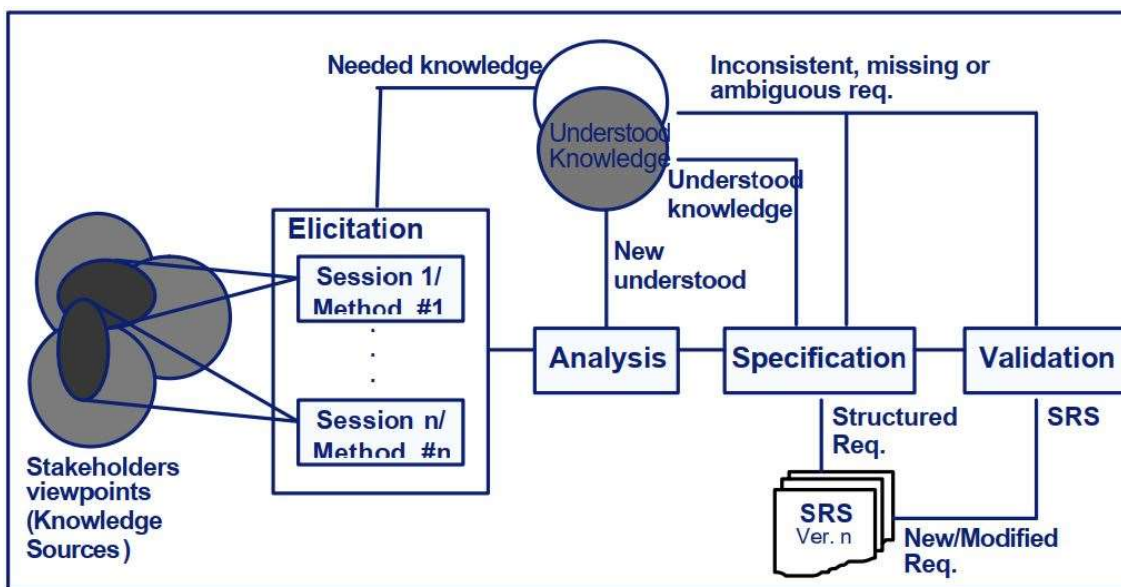
Figure 13 Knowledge and information flows in REP   (Kotonya and Sommerville, 1998)

SECI model for REP

Wan et al. (2010) proposed a process of knowledge conversion in Requirement Engineering Process (REP) based on Nonaka's SECI model (Nonaka and Konno, 1998; Nonaka and Takeuchi, 1995; Nonaka et al., 2000) as shown in Figure 14. This is a single-loop process, in which there are three main players: users, system analysts (requirement engineers and developers). System analysts are experts who comprehensively understand domain knowledge and play a bridging role between users and developers. They design concepts, decompose and integrate requirements for users, while check for consistency, optimize knowledge and flexibility processing for developers. High-level knowledge such as knowledge about users' business and tacit knowledge flows to low-level knowledge such as design specification of developers. Also, developers' software domain knowledge is communicated to users. System analysts bring users' knowledge to developers. Developers deliver requirement specification to users and system analysts and receive feedbacks. These tasks are all executed in a single SECI loop (Wan et al., 2010).
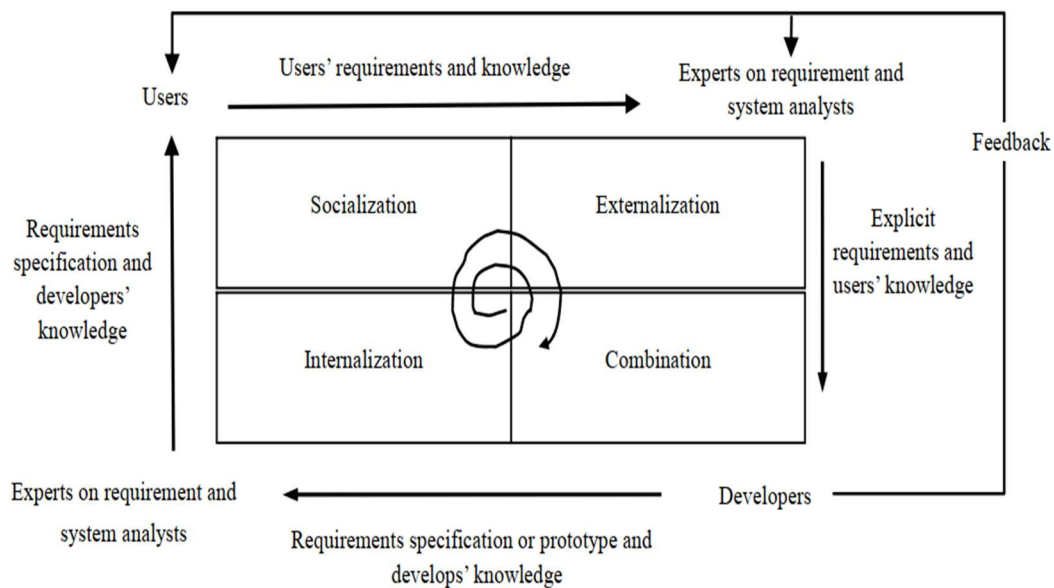
Figure 14 Model of knowledge conversion in REP (Wan et al., 2010)

Knowledge management framework in software RE based on SECI model

Chikh (2011) proposed a knowledge management framework based on the SECI model of knowledge creation in which supporting IT tools such as JAD tools (email, forums,

chat), Domain ontologies, Semantic Wiki, CAD tools, Prototyping tools, Authoring tools are divided into groups and used to construct sub systems of the framework: Socializer, Internalizer, Externalizer, Combiner (Figure 15). This framework is proposed to be used by stakeholders and analysts to facilitate collaboration to exchange and manage knowledge within the software project with the aim is to exploit tacit and explicit knowledge related to software requirements. The framework also attempted to facilitate semantic based interpretation of software requirements by restricting interpretation through domain ontologies. The key point of this framework is to remap the RE activities based on knowledge management viewpoints and combine the SECI model with domain ontologies.
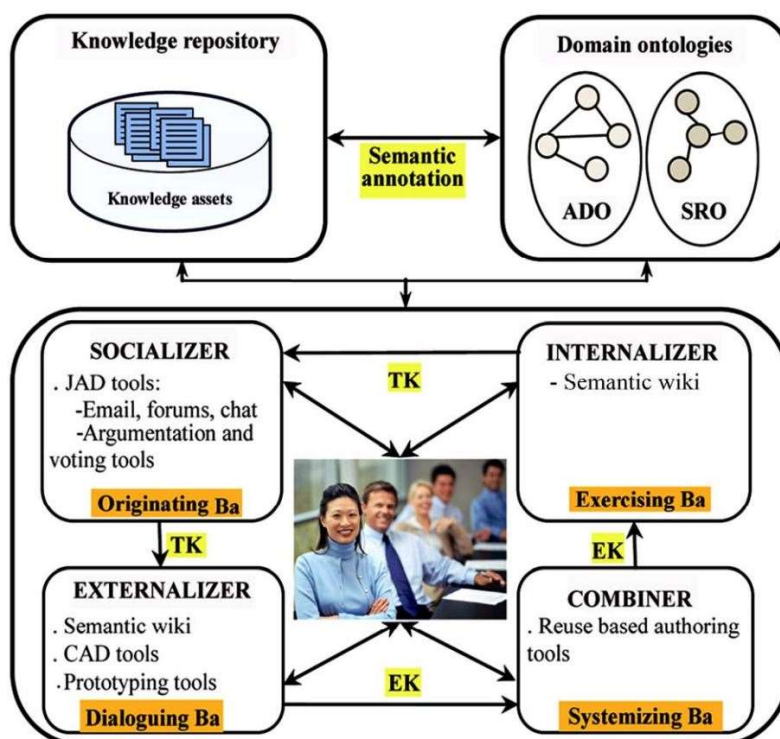
Figure 15 Knowledge management framework for software RE based on the SECI model (Chikh, 2011)

2.6 Ethnography

Ethnographic fieldwork

Ethnographic methods' roots are in sociology and anthropology (O'reilly, 2004). Anthropology is a branch of social science that assembles empirical knowledge and observations to explain the origins and social relationships of human beings.

Ethnography is the informative records of social life and culture in a particular social system based on multiple detailed observations of what people do in the social settings being observed. Ethnographic fieldwork is the key method of anthropology. According to Malinowski (1922), the ethnographer should not stay at the office desk theorizing but should get out, go into the field and spend time experiencing, learning about different peoples/communities in their natural surroundings. Ethnography can be both qualitative and quantitative. Studying specific groups, communities or institutions such as gangs, hospitalized patients, rehabilitating elders etc. are representative examples of ethnography.

Participant observation

The most representative ethnographic method is participant observation. Its core activity is long-term living with the target community. By getting accepted and experiencing daily conduct of the population, the observer gains a sound understanding of local knowledge, values, and rituals as if he/she is a native to the community. It's origins are traced to Malinowski's fieldwork among Trobriand Islanders in 1914 (Malinowski, 1922)(Figure 16). Malinowski was the first to use participant observation to generate specific anthropological knowledge. Wright (2015) emphasized that by living and working in the community, learning and seeing patterns of social behaviors of the community helps the ethnographer to internalize the basic beliefs, fears, hopes, and expectations of the people under study.
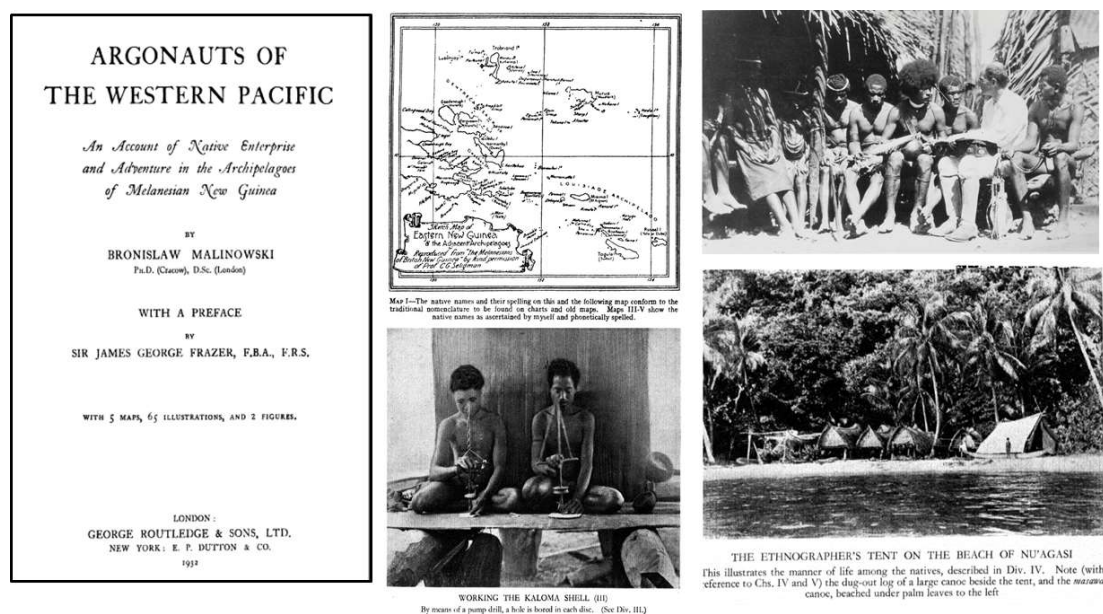
Figure 16 Malinowski's ethnographic fieldworks in Trobriand Island (Malinowski, 1922)

In short, as shown in Figure 17, in participant observation, an investigator (participant observer) studies the life of a community by gaining entrance into, achieving social acceptance by a foreign culture or alien group, sharing in its activities to attain a comprehensive understanding of community.
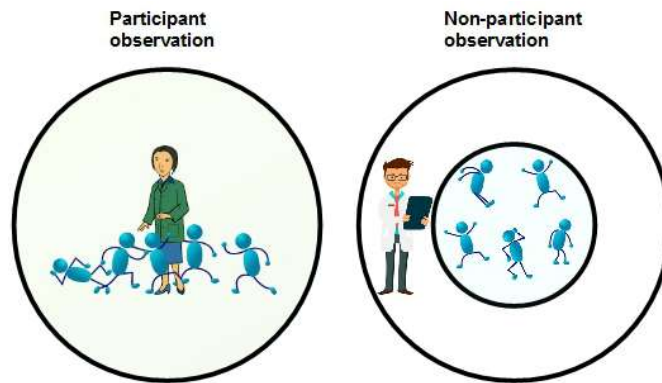
Figure 17 Participant observation vs. Non-participant observation

The results of the participant observation are often recorded in forms of field notes. Field notes are usually hand-written logs, diaries of events, descriptions persons encountered, conversations with local communities. Sometimes depending on the skills or technological tools the ethnographer possesses, there can be artistic sketch, diagrams of buildings, pictures of the locations where the observations occurred. An example of this is shown in Figure 18 (which is the ethnographic field notes of a trip to Mai Chau, Hoa Binh Province, Vietnam in 2005).

(Retrieved from https://hiveminer.com/Tags/ethnography%2Cfieldnotes)
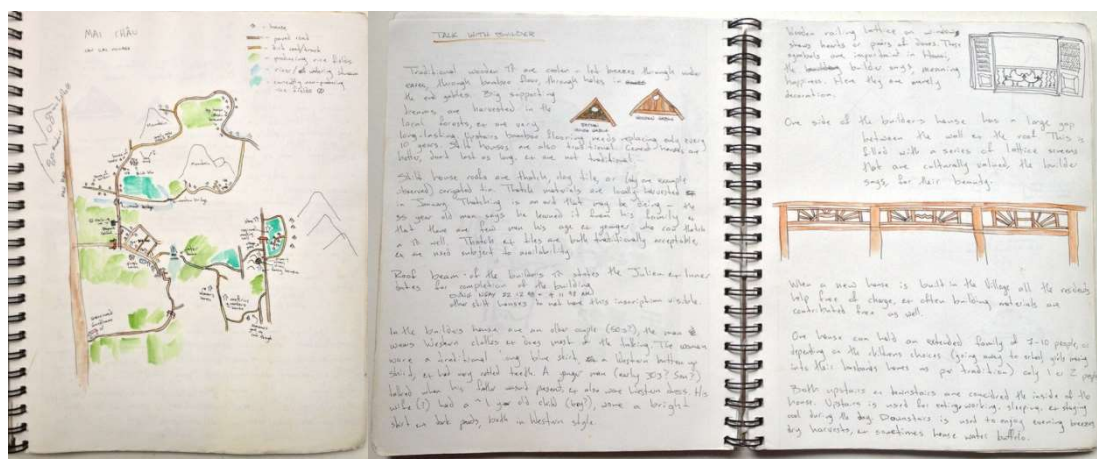
Figure 18 An example of ethnographic field notes

2.7 Contextual Requirement Elicitation

The concept of contextual requirement elicitation

Software requirements are often collected in software vendor's office, at workshops or meetings between customers and requirements engineers. Document analysis, phone discussion with users are also effective method to gather detailed information for requirement formulation. However, in industrial context, software requirements can also be collected directly at the workplace where users use the system. This is the case of manufacturing shop floors that this thesis focuses on. Keller (2011) divided requirement elicitation into 2 types: (1) non-contextual requirements elicitation: elicitation that take places apart from the customer's workplace, (2) *contextual requirements elicitation*: elicitation that take places at the customer's workplace. The main goal of contextual requirement elicitation is to have a firm, clear understanding of the work area of the customer, requirements that cannot be discovered or are easily omitted by conventional elicitation method.

Ethnographic methods for Requirements Elicitation

Manufacturing shop floors have myriad of integration problems and workflow requirements which at times cannot be generalized and peculiar to the target environment only (i.e., situation-specific or context-specific). The elicitation of such requirements must take place at the target manufacturing shop floors or workplaces. This means applying the methodology of section 2.4's ethnography. Onyeka (2013); Viller and Sommerville (2000) listed various ways integrating ethnography into the software engineering. Ethnography is capture system requirements by participatory observation, contextual enquiry. Therefore, ethnography is effective for discovering requirements derived from the way in which people work, rather than the formal processes or standard procedures defined in work-flow instructions. Requirements which are derived from cooperation and awareness of users involved are also effectively discovered by ethnography (Onyeka, 2013). As shown in Figure 19, Viller and Sommerville (2000) proposed an ethnographically informed method for requirements elicitation in which rather than the requirements engineers relying on an ethnographer for analysis of the social nature of the workplace, they now perform the analysis themselves, supported by ethnographically informed guidance which are encapsulations of information about a system or process.
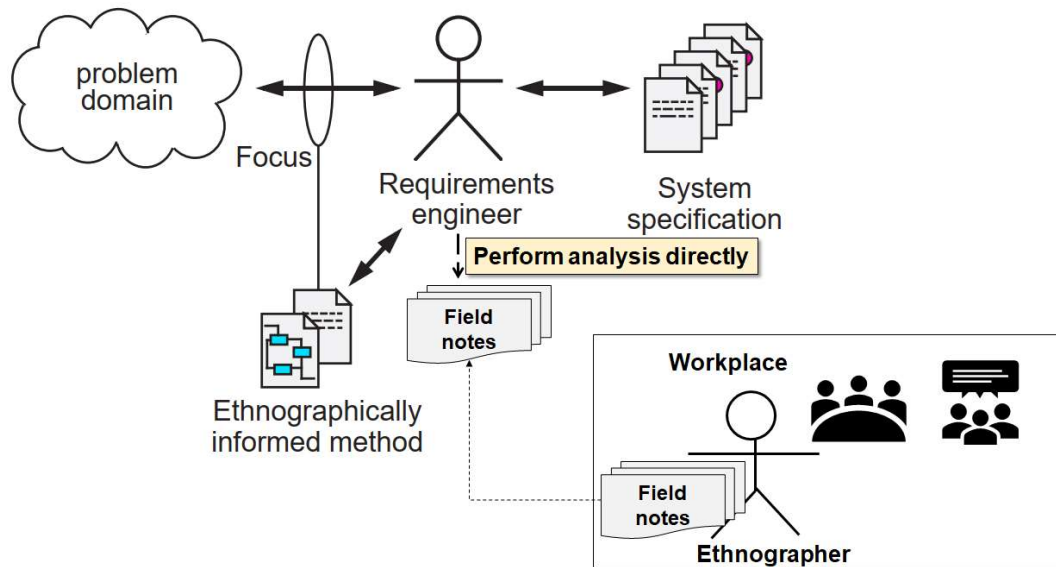
Figure 19 Ethnographically informed method of requirement engineering (Viller and Sommerville, 2000)

2.8 Limitations of previous research

Literature review showed us the following limitations that need to be addressed:

(1) Wan et al. (2010)'s knowledge conversion model assumes requirement engineers who are involved in process are masters who understand thoroughly users' business domain and are trusted by both developers and users which in real life may not necessarily be true. Further, the model assumes developers can give direct feedback on tacit requirements to the users and vice versa, which is also not necessarily valid in case of IT outsourcing and developers' limited access to manufacturing shop floors' or direct contact with shop floor's users due to data security, protection and confidentiality policy of the client's company (Figure 20).(This situation in common in industry settings, more details will be given in the evaluation by case study of this thesis)
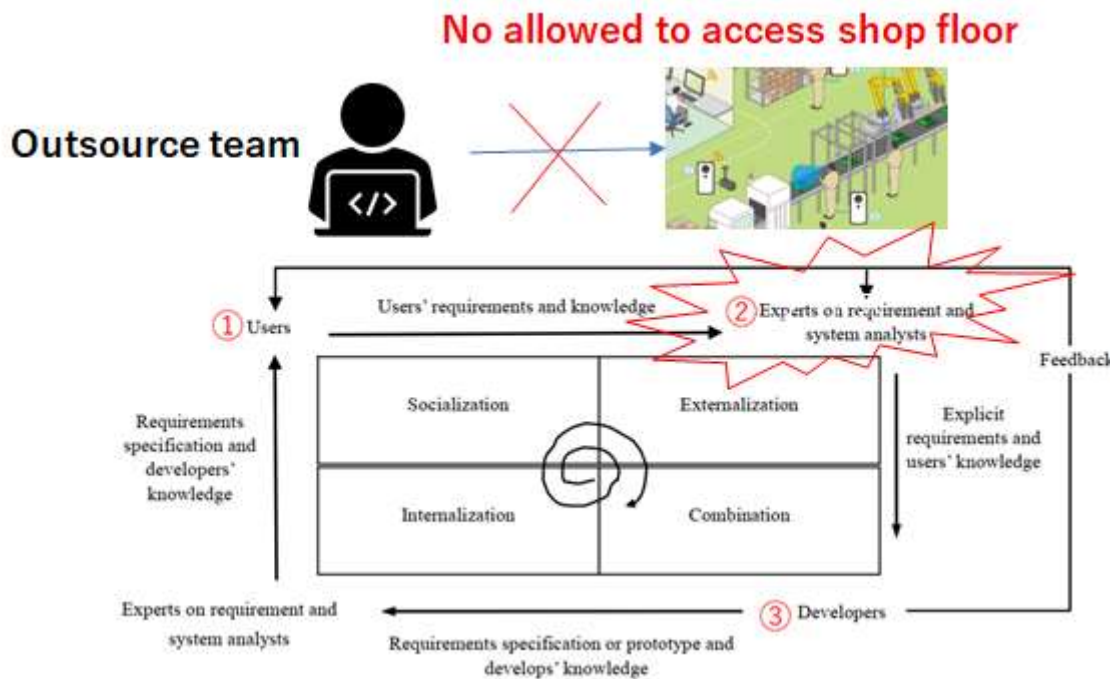
Figure 20 Limitations of the single loop SECI model for REP

(2) Viller and Sommerville (2000)'s ethnographically informed method of requirement engineering process forces the requirement engineers to rely on another third party which is ethnographer which is not effective and might introduce noises to the tacit requirement elicitation process. Furthermore, whether the final two issues: (1) tacit requirement that is never mentioned by the stakeholders/users, (2) inconsistent interpretation of requirements due to ambiguous nature of tacit knowledge possessed by different classes of users) have been solved or not is not visible to the requirement engineer (Figure 21).
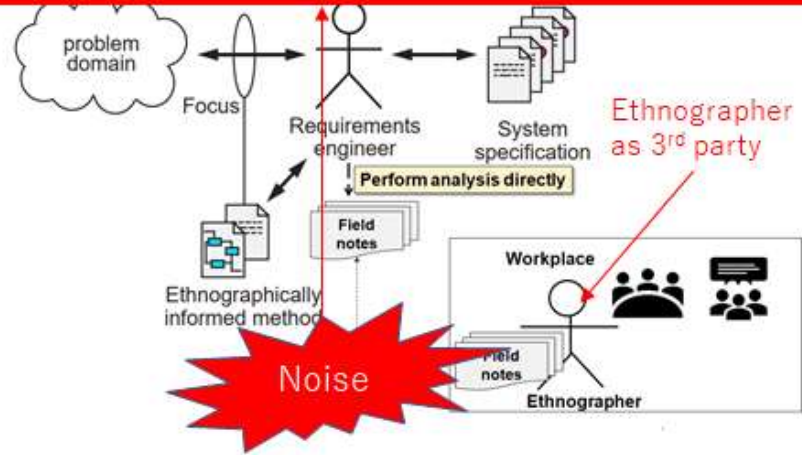
Figure 21 Limitations of ethnographically informed method of requirement engineering
(3) Chikh (2011)'s knowledge management framework is still at a conceptual level. Since knowledge related to software requirements are restricted by predefined domain ontologies, and the four sub systems Socializer, Internalizer, Combiner and especially Externalizer which is in charge of converting tacit knowledge to explicit knowledge heavily rely on IT tools (prototyping, semantic wiki, video recording, questionaires) but not contextual observation, it is difficult to fully capture contextual aspects (tacit requirements) of shop floors. The literature itself also did not elaborate much on how the observations and analysis are performed at the field. The proposed model also implicitly assumed that there is easy access to manufacturing shop floor and effective procedures available for gathering tacit requirements.
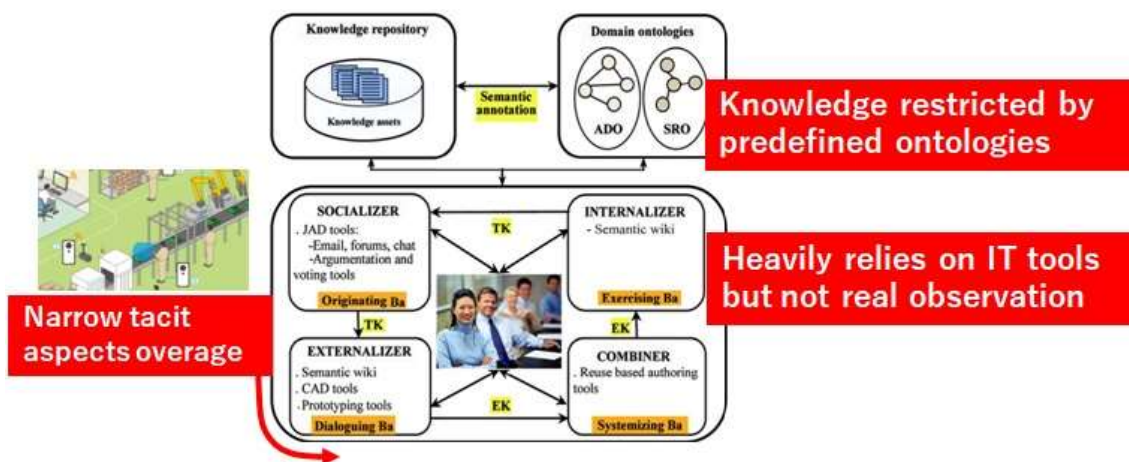
Figure 22 Limitations of (Chikh, 2011)'s SECI-based knowledge management framework

2.9 Summary of literature review

This chapter elaborated on theoretical background of this thesis. With regards to the focus of this thesis - tacit requirements elicitation - the chapter started with an overview of Requirement Engineering was briefly explained with emphasizes on Requirement Engineering process, its role in the whole software development cycle (section 2.1).

Next, fundamental theory of knowledge management where the concept of knowledge, classification of knowledge as well as the well-known SECI model that was first proposed by Nonaka and Takeuchi (1995) to describe how explicit and tacit knowledge is generated, transferred, and recreated in organizations (section 2.2 and 2.3).

To provide readers in-depth understanding of tacit requirements in software engineering, this chapter started by reviewing past researches and listing tacit aspects of software requirements in a systematic manner, similarities between tacit requirements and tacit knowledge. Challenges in tacit requirement elicitations have been pointed with a notice that ambiguity in communication will complicate the process. The above analysis highlights the need of a knowledge-based approach for REP (section 2.4).

The discussion on knowledge-based approach for REP is initialized by reframing the REP in view of knowledge creation. In which, my analysis clearly showed that REP's knowledge creation enablers follow exactly past researches. This is the starting point for modeling REP using Nonaka's SECI model, which is also elaborated by reviewing two representative works of Wan et al. (2010) and Chikh (2011)(section 2.5).

Since the origins of ethnographic methods lie in sociology and anthropology, to help readers have a basic understanding of this methodology, an introduction to ethnography in this field was carried out with explanation on two key concepts: ethnographic fieldwork and participant observation (section 2.6). How ethnography can be useful in requirements elicitation is discussed in section 2.7 of this chapter.

Finally, limitations of previous researches are clearly pointed out in section 2.8.

In the next chapter, a new approach will be proposed to address these limitations.

Chapter 3

Dual loop knowledge conversion model for tacit requirements elicitation

To address the limitations of the previous researches as pointed out in Chapter 2, this research proposed a new model of knowledge conversion. Details of this knowledge conversion model, its merits will be explained in this chapter

3.1 Preliminary analysis

The conclusion of section 2.3.2 showed us the requirements specification is incomplete without considering the physical and organizational environment in which the system will be used. In industrial settings, the two limitations of previous research can be visualized in a summarized form as shown in Figure 23. There are at least two improvements which we need to carry out: (1) First, find a way for requirement engineer to have a more extensive knowledge on manufacturing shop floors more directly with less noise (2) Second, performing ethnography's participant observations in a more effective and knowledge creation enabling manner. Utilizing the concept of "Ba" for SECI model-based knowledge conversion (Nonaka, I., Toyama, R., and Nagata, A. ,2000), from this analysis we can see that there exists 2 "Ba"s: (1) the software development office and (2) the manufacturing shop floors. Therefore, for effective knowledge conversion, it is intuitive to perform participant observations as another SECI loop execution.
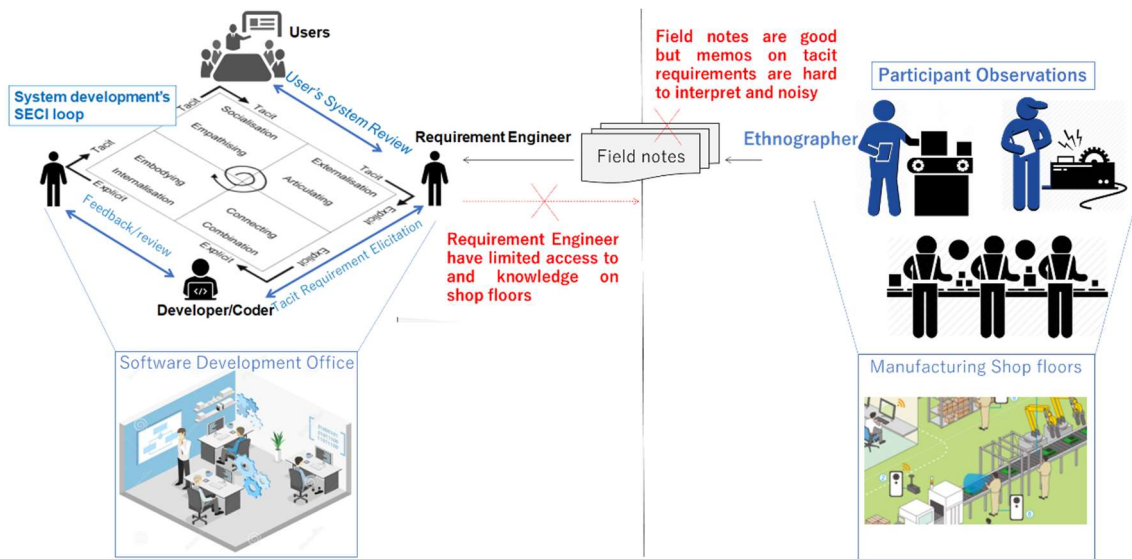
Figure 23 Knowledge conversion constraints of industrial environment

3.2 Dual loop knowledge conversion model

Base on the preliminary analysis at section 3.1, this research proposed a new model of knowledge conversion for requirements engineering process as illustrated in Figure 24. The model has two layers which correspond to two main "Ba"s: (1) The manufacturing shop floor, (2) The software development site.

Lower layer facilitates the Manufacturing shop floor's SECI loop:

The requirement engineer becomes the ethnographer. The requirement engineer as the ethnographer does not just conduct normal observing at work. The requirement engineer spends great amount of time on the shop floor observing, recording, documenting the procedures of how the meisters perform works or operations on the shop floor. Here the observation focused on business processes and operation procedures performed by the production managers, line leaders, machine keepers and operators to carry out production plan ensure smooth progress. At times, he/she performs the operations on the shop floor him/herself to grasp the internal thinking of meisters that lead to tacit requirements.

Upper layer facilitates the System development SECI's loop:

This layer has SECI loop run through a similar process as Wan et al. (2010)'s knowledge conversion model. However, this model has two modifications. Firstly, the requirement engineers who are also the ethnographer utilize his own field notes for tacit requirements elicitation. Secondly, there is no direct interaction between users and

developers, since the two parties can now rely on the facilitator who the requirement engineers who have gained domain expert knowledge through ethnography.

It is important to bear in mind that at each layer, the SECI conversion process can be executed as many times as possible to refine the requirements as well as to facilitate the development process.
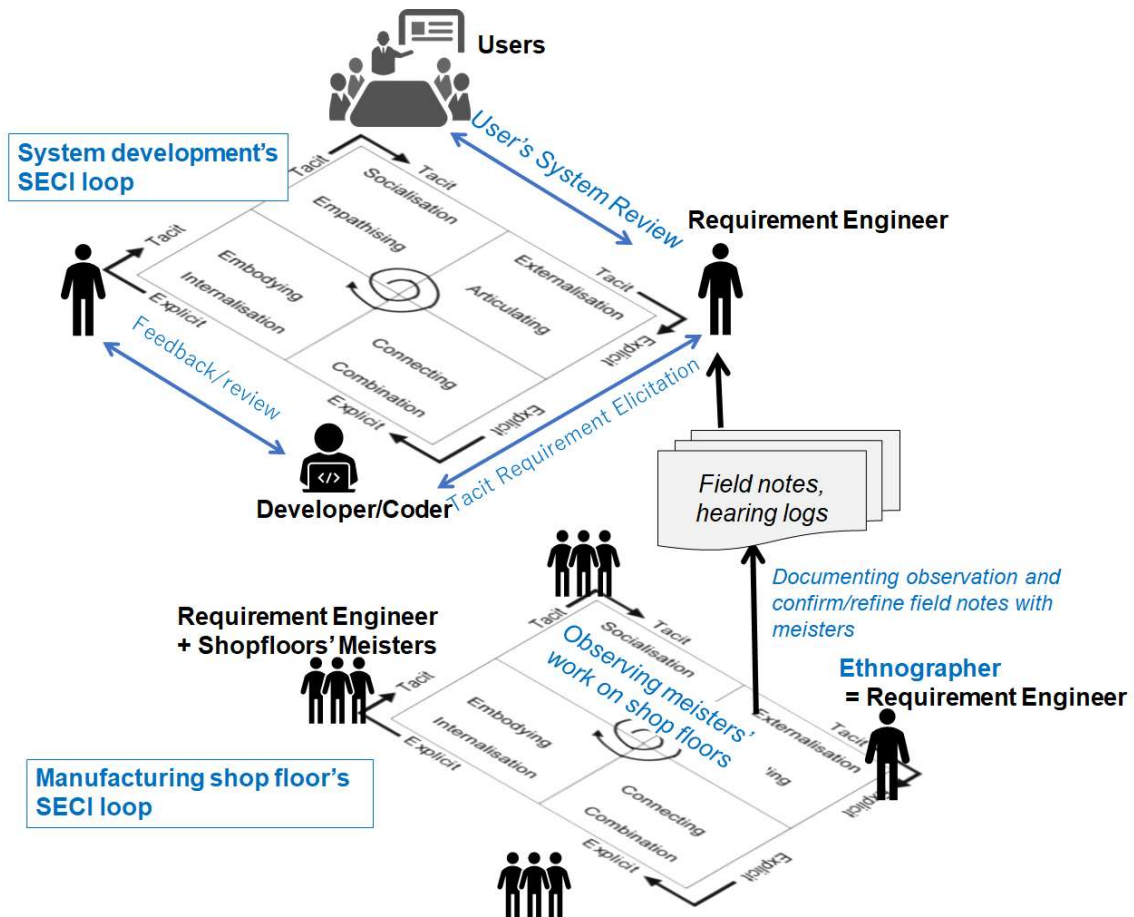
Figure 24 Dual loop knowledge conversion model

The merits of this dual loop conversion model are as follows:

Firstly, at Manufacturing shop floor's SECI loop the requirement engineer does not have to be masterful of users' business domain and trusted by both developers and users in the first place, which eliminates Wan et al. (2010)'s knowledge conversion model's constraints

Secondly, the issues mentioned in section 2.8 are addressed. Tacit requirements which are never mentioned by the stakeholders/users are expected to be fully discovered

41

through ethnographical work of the requirement engineer. Inconsistent interpretation of requirements (due to ambiguous nature of tacit knowledge possessed by different classes of users) is eliminated by externalized and documented observations. This helps overcome the problem of third party's noise of Viller's ethnographically informed method.

Thirdly, System development SECI's loop ensure smooth development process even in case of IT outsourcing where developers have no direct contact with users due to data security, protection and confidentiality policy of the client's workplace (as specified in in the case study).

3.3 Summary

In this chapter, we have proposed a new knowledge conversion model – The dual loop knowledge conversion model - which integrates ethnography and the knowledge creation process (especially the process of conversion from tacit knowledge to explicit knowledge) to fully capture and implement tacit requirements at manufacturing shop floors. In this model, the roles of the requirement engineer and the ethnographer are combined into one. Ethnographer's participant observation

Chapter 4

Evaluation (Case study)

The effectiveness of the newly proposed method was evaluated and verified by analyzing the following case study.

4.1 Background information

The proposed model has been applied to the development of a scheduling/dispatching system for integrated fab automation of a semiconductor company in Japan (Figure 25).

Figure 25 Scheduling/dispatching system for integrated fab automation

Organization chart of the system development project is shown in Figure 26. The system is to be installed for a front-end process clean room with 24 hours (3 shifts) operation. The clean room has total area of 8,000m^2. There about 800 operators including line leaders who work in this clean room to deliver about product 20 or more categories of product. On average, the number of process steps is 200~300, the number of machines is in order of 100s. System developers are outsourced to on-site IT vendors. Although being on-site these vendors have no direct access to the clean room or direct

contact with users due to information security and confidentiality policy. Therefore, all system development for the clean room is in performed under the supervision of the factory's manufacturing engineering department. This department has system analysts as core members will perform requirement engineering, system delivery as well as controlling and keeping track of project progress from system vendors.
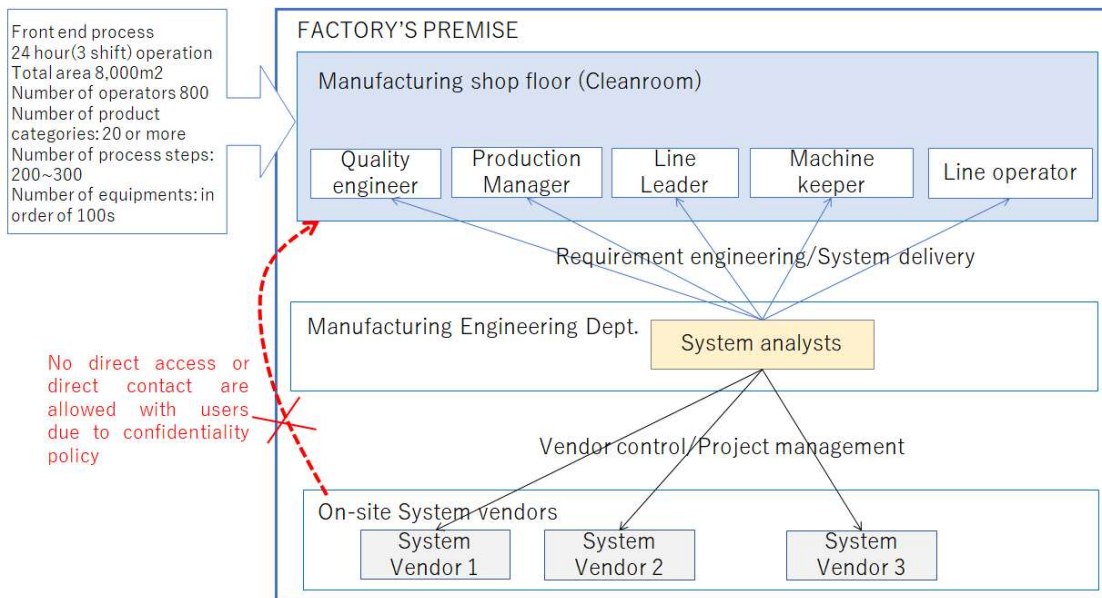
Figure 26 Organization chart of system development team
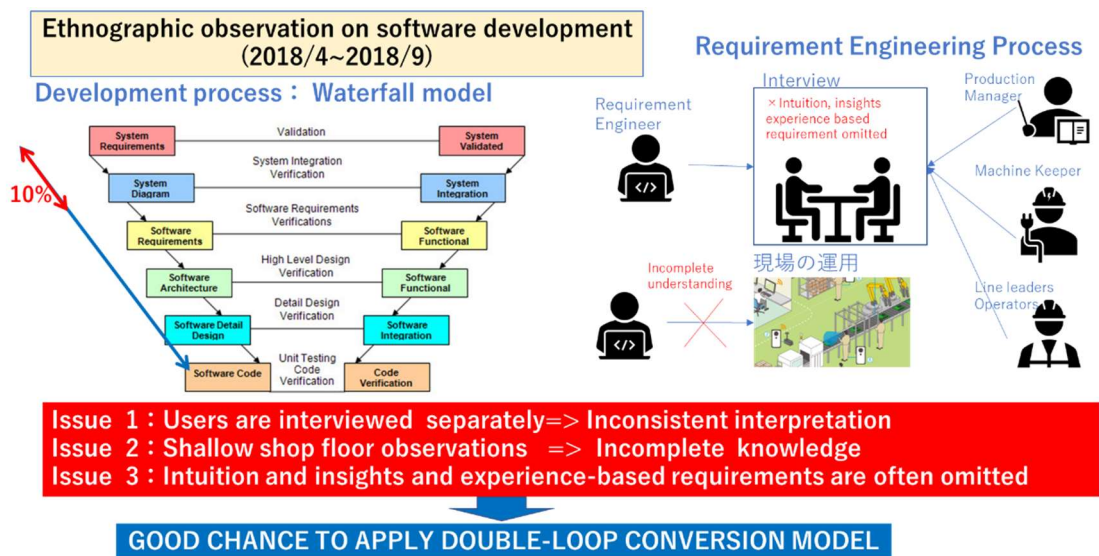
Figure 27 Participant observation on software development process at the factory

Furthermore, ethnographic observation on software development (Figure 27) found out that the factory has a long time frame and carefully planned budget for the project. Also, complex integration with other sub-systems in the manufacturing system make the

44

factory to adopt V-model software process for sophistication. However, requirements engineering seemed to not have enough attention (only 10% of total man-hour). While the is a wide range of related stakeholders (line leaders, machine keepers, operators etc.), each group is interviewed separately, which led to inconsistent interpretation of tacit requirements. System vendors' engineers limited access to shop floors induced shallow shop floor observations. Hence vendors on their own have incomplete knowledge about clean room. Intuition and insights and experience-based requirements are often omitted during interview. These characteristics make this case study provides a perfect example for the verification of the dual loop knowledge conversion model proposed due to the following reasons: (1) The target clean room have diversified product line-ups and processing conditions, for which skillful operators or line leaders have gather certain amount of experience and know-how (tacit requirements) to control the production smoothly, (2) Developers have no direct access to manufacturing shop floor or contact with users.

4.2 Experimentation

In this case study, the ethnographer and the requirement engineer are the same person. In the Manufacturing shop floor's SECI loop - the ethnographer/requirement engineer collaborated closely with clean room's operators, line leaders and machine keepers to elicit tacit requirements (Figure 28). The ethnographer/requirement engineer spent totally 3-5 weeks to first enter the clean room and perform participant observations of how line operators, line leaders manually dispatch work to discover tacit rules of operations - Socialization. Observations are then documented into business requirement specification drafts, field notes and hearing logs – Externalization (converting tacit knowledge to explicit knowledge). After combining observations, the ethnographer further categorized and summarized requirements through review meetings with clean room's related users - Combination. Finally, the ethnographer ran a simple simulation of dispatching rule as a way of applying observed ideas - Internalization.

Manufacturing shop floor's SECI loop

	Tacit knowledge	Tacit knowledge
Socialization	Observing line operator, line leaders manual dispatch work in clean room	
Externalization		Documenting observation into business requirement specification, field notes, hearing logs
Internalization	Run simple simulation of dispatching rule as a way of applying observed ideas.	
Combination		Combine observations, categorize and summarize requirements through review meetings with users

Line Leader / Machine keeper / Line operator

System analysts

Figure 28 Manufacturing shop floor's SECI loop of the case study

Figure 29 shows how the System development's SECI loop is executed in this case study. First, based on the first SECI loop's field notes and hearing logs, system analysts elicit tacit requirements into requirement specifications then developers implement system functions based on this – Combination. Developers internalize this knowledge by building system prototype, performing test and get it reviewed by system analysts – Internalization. On the other half of the loop, since developers (system vendors) have no direct contact with users, system analysts will receive the system from vendors, have the users pilot run the newly developed system and give feedback through user review meeting – Socialization. System analysts document review results into bug reports, issues log etc. for further feedback to developers – Externalization.
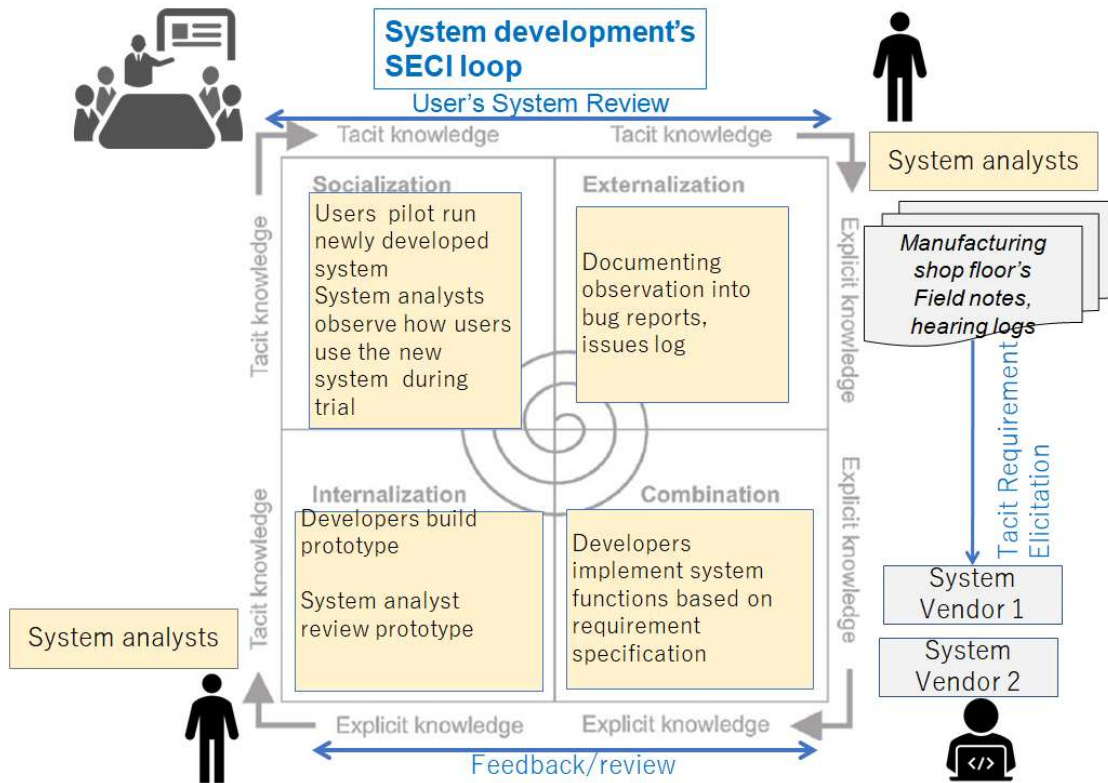
Figure 29 System development's SECI loop of the case study

Tacit requirements elicitation in this case required the conversion of how the experienced operators in clean room perform complex operations which required experience such as process condition tuning of ion implantation, metal sputtering for the highest production throughput. Depending on the current condition of the equipment or the recipe being used, scheduling the next wafer lot must consider this process tuning loss.

4.3 Enhancement of participant observation techniques for tacit requirements elicitation

Howell (1973) divide participant observation into 4 phases as shown in Table 5. Applying the procedures suggested by the 4 phases of participant observations, as the ethnographer/requirement engineer I took initiative and made some enhancements in participant observation techniques to execute the proposed knowledge conversion model more effectively and elicit tacit requirements better at the shop floor as shown in Table 6. A sample ethnographic analysis of narratives and recordings for tacit requirements is display in Table 7

Table 5 Four phases of participant observation

Phases of participant observation	Description
1) Build a relationship of mutual understanding or trust and agreement between people	Howell (1973) emphasized the importance of becoming friends, get accepted in the community to obtain quality data.
2) Performing activities in the field	DeWalt, DeWalt, and Wayland (1998) emphasized the importance of "talking the talk" and "walking the walk", do as the locals do to connect or show a connection with the community.
3) Recording observations and data	Write down the observations, taking the pictures, saving the data gathered
4) Analyzing data	Sorting information gathered, dividing and grouping recorded conversations into common themes, and constructing a coherent story from data

Table 6 Enhancement of participant observation techniques

Phases of participant observation	Efforts made by the author for participant observations during case study
1) Build a relationship of mutual understanding or trust and agreement between people	Wear clean room gown, enter the clean room early in the morning to attend the morning meeting, shift change meetings. Take over simple dispatching works for line leaders/operators to have more short breaks. Have lunch together with line leaders, operators, machine keepers. Take a break together and chat with production managers, line leaders, operators at smoking room
2) Performing activities in the field	Do as the locals do: Wear clean room gown, go into the clean room. Do not just observe and record data but also do the dispatching

	work together with line leaders and operators like a real operator
3) Recording observations and data	Write down the observations into log book (Figure 30), taking the pictures, saving the data gathered of the field work
4) Analyzing data	Focus on recorded parameters of machines, procedures (steps and sequences) of dispatching work at the target machines. Analyze narratives of instructions to operators from line leaders, phone conversation from production managers to line leaders. Categorizing information gathered through recorded conversations, interviews, finding common themes, and constructing a coherent story from data

*Field notes are record in blue paper notebook designed only to be used for clean room
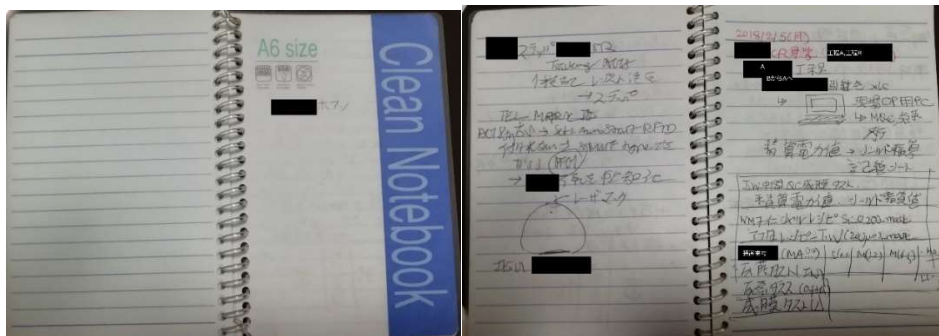
Figure 30 An excerpt field notes recorded during the case study

Table 7 Sample ethnographic analysis of narratives and recordings for tacit requirements

SAMPLE NARRATIVE ANALYSIS
NARRATIVES: 2018/2/6 07:55 observations of dispatching work at process P
Line leader M: Lot L387 has been waiting for start so long, maybe since last night. Why?
Operator K: The machine M-1 has reached threshold for parameter Y. We had to stop dispatching L387 to M-1 and waiting for machine maintenance.
Line leader K: I see. Let's call machine keeper T. What is the product type of L387?
Operator K: Product type ZK. Is there any substitution machine?
Line leader K: This is specialized product. There is no substitution. We just have to wait.
MACHINE'S PARAMETER RECORDINGS (in ethnography field notes)
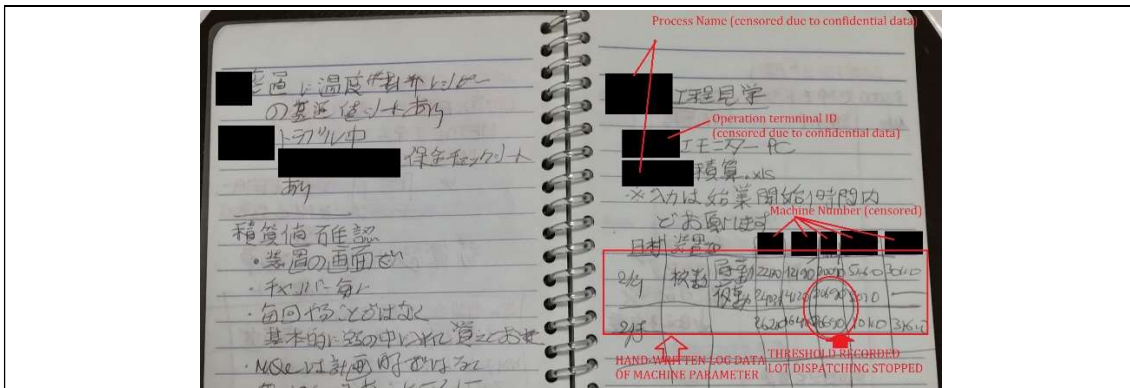
INTERVIEWS:
Requirement Engineer: Based on my observations, it seems you forgot to machine's parameter Y which will increase overtime and affect the availability of machine when reach certain threshold value.   Line leader M: I forgot to mention our electrical power auto adjusting system. I'm so familiar to the fact it is always running in the background as a routine. Therefore, I do not even notice it or have it in my mind anymore
TACIT REQUIREMENTS: Machine's parameter Y which will increase with the number of dispatched works increasing. Stop the dispatching instructions if $Y \geq$ Maximum Threshold Y_{max}

4.4 Evaluation Metrics

The effectiveness of the method was measured using the following metrics: Total number of use cases discovered, Number of tacit requirements not discovered, Lead time for requirement engineering process, Total system development lead time, System pilot run result. We measured and compared the outcome of each metric before and after applying the proposed dual loop conversion model.

Table 8 Evaluation metrics

METRICS
(1) Total number use cases : Expected to be maximized
(2) Number of tacit requirements not discovered: Expected to be 0
(3) Total development lead time : Expected to be reduced
(4) System launch result : Success is a MUST

4.5 Evaluation Results

The result of the case study is shown in Table 9. As expected, with the dual loop tacit requirements elicitation process, the total use cases discovered increased from 35 to 75, of which 40 uses cases were omitted or unable to be discovered when now knowledge

conversion process was applied. System pilot run also achieved positive result which led to official launch of the system.

Table 9 Evaluation Results

	BEFORE	AFTER
Number use cases discovered	35	75
Number of missing tacit requirements	40	0 [a]
Lead time for requirement engineering process	2 months	4 months
Total system development lead time	18 months	10 months
System pilot run result	Trial use stopped after just one month due to mismatch with shop floor workflow	System accepted by the users and launched officially

[a] while there might be more undiscovered tacit requirements, consensus of manufacturing shop floor is "when system requirements are verified to ensure basic level of operation with no problems, this can be considered to be zero."

Because substantial amount of time spent on the ethnographical work, the lead time for requirement engineering process doubled from 2 months to 4 months. However, fully discovered requirements helped reduce overall development process from 18 months to 8 months. This is due to less rework at later stages (designing, coding and testing) to cover missing or inconsistent requirements.

The 40 use cases of tacit requirements are categorized in to (1) requirements related to experience in maintenance of processing equipment; (2) requirements related to know-how of product type and equipment compatibility; (3) requirements related to know-how to process recipes switching loss and (4) requirements derived from cooperation and awareness of production output of previous process/shop in the same line as shown in Table 10.

Table 10 Case Study's Tacit Requirements Categories

Category	Number of use cases
Requirements related to experience in maintenance of processing equipment	20
Requirements related to know-how of product type and equipment compatibility	5
Requirements related to know-how to process recipes switching loss	5
Requirements derived from cooperation and awareness of production output of previous process/shop in the same line	10

4.6 Summary

In this research, we proposed a dual loop model to overcome the shortcomings of previous works. In this chapter this model is verified and evaluated by performing a case study of "Developing the scheduling/dispatching system for integrated fab automation of a semiconductor company in Japan". Case study findings are in the following: Missing tacit requirements were fully discovered. High quality requirements specification was produced, and the project achieved a successful launch of the system. Due to time-consuming nature of ethnography process, lead time for requirement engineering process doubled from 2 months to 4 months, but total development lead time is shortened by 8 months thanks to less rework at later stages.

Chapter 5

Conclusion

In knowledge-intensive environment, especially manufacturing shop floors, tacit requirements are crucial to software development. However, tacit requirements are hard to be discovered, elicited and codified. In this research, after reviewing past research works which utilized the SECI model of knowledge conversion for requirement engineering process and ethnography for contextual requirement elicitation, we proposed a dual loop model to overcome the shortcomings of previous works. This model is evaluated and proven effective by the case study detailed in chapter 4. To conclude this thesis, in this chapter the answers to subsidiary research questions, main research questions are summarized.

5.1 Answers to subsidiary research questions

SRQ1: *What are the challenges of acquiring tacit requirements at the manufacturing shop floors?*

Answer to SRQ1: There are four issues which hinder the process of tacit requirement elicitation: (1) tacit requirement that is not made explicit by stakeholders/users but presumably implied in requirements by users, (2) tacit requirement that is never mentioned by the stakeholders/users, (3) inconsistent interpretation of requirements due to ambiguous nature of tacit knowledge possessed by different classes of users,(4) On-site system engineers (who are outsourced vendors) have limited access to manufacturing shop floors due to data security, protection and confidentiality policy of the company, therefore useful participant observations are near impossible.

SRQ2: *What role does the knowledge creation process (especially the tacit-to-explicit knowledge conversion process) play in acquiring tacit requirements?*

Answer to SRQ2:

Analysis in this research showed that knowledge creation process has 3 roles in tacit requirements development

Role 1: Tacit requirement elicitation initializer - The knowledge conversion process from tacit knowledge to explicit knowledge is indispensable for acquiring implicit requirements with the focus is on "Requirement Elicitation" step to translate on-site

know-how into system requirements.

Role 2: Know-how organizer - The knowledge conversion process from tacit knowledge to explicit knowledge systematizes on-site know-how based on long experience and intuition so that the corresponding requirements can be expressed comprehensively

Role 3: Consistency enforcer - The knowledge conversion process from tacit knowledge to explicit knowledge aligns the interpretation and understanding of the tacit requirements of various stakeholders in across departments to eliminate discrepancies, which eventually help to create the final requirements specification and perform verification effectively.

SRQ3: *What is an effective process for capturing implicit requirements in knowledge-intensive manufacturing shop floors?*

A dual loop knowledge conversion model - which integrates ethnography and the knowledge creation process (especially the process of conversion from tacit knowledge to explicit knowledge) is necessary to fully capture and implement tacit requirements at manufacturing shop floors. The model has two layers which correspond to two main "Ba"s: (1) The manufacturing shop floor, (2) The software development site.

Lower layer facilitates the Manufacturing shop floor's SECI loop:

The requirement engineer becomes the ethnographer. The requirement engineer as the ethnographer does not just conduct normal observing at work. The requirement engineer spends great amount of time on the shop floor observing, recording, documenting the procedures of how the meisters perform works or operations on the shop floor. Here the observation focused on business processes and operation procedures performed by the production managers, line leaders, machine keepers and operators to carry out production plan ensure smooth progress. At times, he/she performs the operations on the shop floor him/herself to grasp the internal thinking of meisters that lead to tacit requirements.

Upper layer facilitates the System development SECI's loop:

This layer has SECI loop run through a similar process as (Wan et al., 2010) knowledge conversion model. However, this model has two modifications. Firstly, the requirement engineers who are also the ethnographer utilize his own field notes for tacit

requirements elicitation. Secondly, there is no direct interaction between users and developers, since the two parties can now rely on the facilitator who the requirement engineers who have gained domain expert knowledge through ethnography

5.2 Answers to the main research question

Main research question (MRQ):

In knowledge-intensive manufacturing shop floors, what characterize an effective knowledge conversion model for tacit requirements elicitation?

An effective process of tacit requirements elicitation at knowledge-intensive manufacturing shop floors should take great consideration of knowledge conversion process that runs in background of this. The working environment for requirements engineering at manufacturing shop floors should make sure Von Krogh et al., (2000)'s five knowledge creation enablers: (1) Instill a knowledge vision, (2) Manage conversations, (3) Mobilize knowledge activists, (4) Create the right context, and (5) Globalize local knowledge are satisfied. This can be obtained by lowering the both the communication barrier between developers and shop floors' stakeholder. At the same, physical barrier between software development sites and shop floors should also be eliminated so that useful ethnographic observations can be conducted facilitating the externalization of shop floors' tacit knowledge. To implement this, this research proposed a new knowledge conversion model – The dual loop knowledge conversion model - which integrates ethnography and the knowledge creation process (especially the process of conversion from tacit knowledge to explicit knowledge) to fully capture and implement tacit requirements at manufacturing shop floors. In this model, the requirement-engineer and the ethnographer are merged into one. Ethnographer's participant observation proved to be helpful for the requirement engineer to fully capture tacit requirements at knowledge-intensive manufacturing shop floors

5.3 Theoretical Implications

This research has focused on requirements engineering in knowledge-intensive manufacturing shop floors where tacit requirements are very difficult to elicit. In today's highly competitive and rapidly changing business environment, shop floors' line leaders and operators are no more permanent employees who hold the knowledge of shop floors until retirement and transfer it accordingly to the next generations of employees. Gradually, outsourced labors such as part time workers, sub-contractors replace this source of labors. For any software system, especially manufacturing systems, system

requirements and specifications are representations of knowledge on shop floors' workflows, standard procedures and processes. Hence, approaching requirements engineering in this context based on Nonaka and Takeuchi (1995)'s knowledge creation theory is intuitive and justifiable. In such cases, how we address the bottle neck of such requirements engineering process which is tacit requirement elicitation will significantly affect the quality as well as the results of requirements engineering work. While contextual requirement elicitation or ethnography has been investigated as early as in the begin (Bentley et al., 1992; Harper, 2000; Hughes, King, Rodden, and Andersen, 1994; O'Brien, Rodden, Rouncefield, and Hughes, 1999; Viller and Sommerville, 2000), these studies made assumptions that did not fully take into consideration constraints of industrial environments which hinder the observations of the ethnographer and useful utilization of field notes.

This research has some theoretical implications for requirements engineering and knowledge management in software projects.

The first theoretical contribution of this research is the proposal of a new way to apply ethnography in requirements engineering in view of knowledge creation. This provides an opportunity for an interaction between contextual enquiries and requirements engineering.

Secondly, we have demonstrated in this study that, failing to consider environment constraints of organizational structure especially the relative positions of manufacturing shop floors with regards to the software development teams will reduce significantly the effectiveness of the knowledge co-creation process that helps tacit requirements elicitation.

Thirdly, evaluation results in the case study one more time confirmed the universal correctness of Nonaka and Takeuchi (1995)'s knowledge creation theory – the SECI model. Furthermore, we showed that in a very knowledge-intensive environment like manufacturing shop floors, there are many "Ba"'s to create new knowledge for requirements engineering. However, separated and individual execution of the SECI loop of each "Ba" posed some limitation on tacit requirements elicitation. This research highlighted that facilitating and maintaining connectivity between these seemingly separated SECI loops is crucial to the requirements development process.

5.4 Practical Implications

To minimize the fluctuations in demand or economic downturns and to reduce labor cost, companies resort to low-cost outsourcing or hiring contract operators. Manufacturing shop floors are always knowledge-intensive environment where highly skilled workforce with their strong tacit knowledge. This trend will continue in future. The proposed model in this research with a favorable verification results from a real-life case study has been proven effective. Managers at manufacturing enterprise and IT vendors can take this as a reference to customize and deploy a similar knowledge-conversion model at their own or their clients' respective shop floors to improve the quality as well as reduce cost of software development of manufacturing systems.

5.5 Future works

Due to limited scope of this study, this research has the following three limitations which can be addressed in future

Reducing lead time of shop floors observations

As specified in chapter 4, participant observations on shop floors takes a long time to complete (as in the case study of this research lead time for REP doubled from 2 months to 4 months). This is because the ethnographer has to spend enough time experiencing the real operations of the workplace to understand the process, to internalize unspoken rules and to collect enough amount of data so that there are no omissions. This posed a problem of reducing the lead time of shop floors observation.

Reducing dependence on skill levels of the ethnographer

The case study might induce some question on how the result of participant observations may depend on the skill levels of the ethnography. If the requirement engineer who performs ethnographic study has some expertise on the related domain of the shop floors, it is easier for him/her to connect events observed, analyzing textual description, recorded data. Furthermore, in the next step, translating these analysis results into software requirements depends on the requirement documenting skills of the requirement engineer. So far, there is a systematic approach to this issue. Future works may look into classifying ethnographic works with regards to target domain and shop floors, forming standard procedures and building IT tools to support the ethnographer.

Expanding the scope of the research

The interviews and participant observations were conducted on small scale and specialized case study – semiconductor factory. In future it could be possible to increase the sample size of population (subjects), varieties of manufacturing shop floors for more generalization of results. Since our study was a confirmatory, that compares the finding of literature study and interview results. In this thesis, only issues in requirement elicitation process were focused, it could also be possible to deal with other adjacent activities like requirement analysis, specification, documentation and validation in future.

References

Akiyoshi, M. (2008). Knowledge sharing over the network. *Thin Solid Films*, *517*(4), 1512–1514.

Alavi, M., & Leidner, D. E. (2001). Knowledge management and knowledge management systems: Conceptual foundations and research issues. *MIS Quarterly*, 107–136.

Anderson, D. J. (2010). *Kanban: successful evolutionary change for your technology business*. Blue Hole Press.

Argote, L., & Ingram, P. (2000). Knowledge transfer: A basis for competitive advantage in firms. *Organizational Behavior and Human Decision Processes*, *82*(1), 150–169.

Basir, B., & Salam, R. (2015). Tacit requirements elicitation framework. *ARPN J. Eng. Appl. Sci*, *10*(2), 572–578.

Bell DeTienne, K., Dyer, G., Hoopes, C., & Harris, S. (2004). Toward a model of effective knowledge management and directions for future research: Culture, leadership, and CKOs. *Journal of Leadership & Organizational Studies*, *10*(4), 26–43.

Bentley, R., Hughes, J. A., Randall, D., Rodden, T., Sawyer, P., Shapiro, D., & Sommerville, I. (1992). Ethnographically-informed systems design for air traffic control. *CSCW*, *92*, 123–129.

Berry, D. M., & Kamsties, E. (2004). Ambiguity in requirements specification. In *Perspectives on software requirements* (pp. 7–44). Springer.

Boehm, B. W. (1984). Software engineering economics. *IEEE Transactions on Software Engineering*, (1), 4–21.

Bostrom, R. P. (1989). Successful application of communication techniques to improve the systems development process. *Information & Management*, *16*(5), 279–295.

Chikh, A. (2011). A knowledge management framework in software requirements engineering based on the SECI model. *Journal of Software Engineering and Applications*, *4*(12), 718.

DeWalt, K. M., DeWalt, B. R., & Wayland, C. B. (1998). Participant observation. *Handbook of Methods in Cultural Anthropology*, 259–300.

Forsberg, K., & Mooz, H. (1991). The relationship of system engineering to the project cycle. *INCOSE International Symposium*, *1*(1), 57–65. Wiley Online Library.

Frederick P. Brooks, J. (1987). No silver bullet essence and accidents of software

engineering. *Computer*, (4), 10–19.

Harper, R. H. R. (2000). The organisation in ethnography–a discussion of ethnographic fieldwork programs in CSCW. *Computer Supported Cooperative Work (CSCW)*, *9*(2), 239–264.

Howell, J. T. (1973). *Hard living on Clay Street: Portraits of blue collar families* (Vol. 956). Anchor Books.

Hughes, J., King, V., Rodden, T., & Andersen, H. (1994). Moving out from the control room: ethnography in system design. *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, 429–439. ACM.

IEEE 610.12. (1990). *IEEE standard glossary of software engineering terminology*. Retrieved from http://ieeexplore.ieee.org/servlet/opac?punumber=2238

Ismail, S., & Ahmad, M. S. (2011). Emergence of social intelligence in social network: A quantitative analysis for agent-mediated PKM processes. *ICIMU 2011: Proceedings of the 5th International Conference on Information Technology & Multimedia*, 1–7. IEEE.

Janz, B. D., & Prasarnphanich, P. (2005). Understanding knowledge creation, transfer, and application: Investigating cooperative, autonomous systems development teams. *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, 248a-248a. IEEE.

Joe, C., Yoong, P., & Patel, K. (2013). Knowledge loss when older experts leave knowledge-intensive organisations. *Journal of Knowledge Management*, *17*(6), 913–927.

Kagermann, H. (2015). Change through digitization—Value creation in the age of Industry 4.0. In *Management of permanent change* (pp. 23–45). Springer.

Keller, T. (2011). Contextual requirements elicitation. *Seminar in Requirements Engineering, Spring 2011, Department of Informatics*.

Kothari, C. R. (2004). *Research methodology: Methods and techniques*. New Age International.

Kotonya, G., & Sommerville, I. (1998). *Requirements engineering: processes and techniques*. Wiley Publishing.

Ladas, C. (2008). SCRUMBAN, And Other Essays on Kanban Systems for Lean Software Development. A Division of Modus Cooperandi. *Inc.–Seattle, USA*.

Loebbecke, C., van Fenema, P. C., & Powell, P. (2016). Managing inter-organizational knowledge sharing. *The Journal of Strategic Information Systems*, *25*(1), 4–14.

Mahr, D., & Lievens, A. (2012). Virtual lead user communities: Drivers of knowledge creation for innovation. *Research Policy*, *41*(1), 167–177.

Malinowski, B. (1922). Argonauts of the Western Pacific. EP Dutton & Co. *Inc., New York*.

Neta, R., & Pritchard, D. (2009). *Arguing about knowledge*. Routledge.

Nonaka, I., Kodama, M., Hirose, A., & Kohlbacher, F. (2014). Dynamic fractal organizations for promoting knowledge-based transformation–A new paradigm for organizational theory. *European Management Journal*, *32*(1), 137–146.

Nonaka, I., & Konno, N. (1998). The concept of "Ba": Building a foundation for knowledge creation. *California Management Review*, *40*(3), 40–54.

Nonaka, I., & Takeuchi, H. (1995). *The knowledge-creating company: How Japanese companies create the dynamics of innovation*. Oxford university press.

Nonaka, I., Toyama, R., & Nagata, A. (2000). A firm as a knowledge-creating entity: a new perspective on the theory of the firm. *Industrial and Corporate Change*, *9*(1), 1–20.

O'Brien, J., Rodden, T., Rouncefield, M., & Hughes, J. (1999). At home with the technology: an ethnographic study of a set-top-box trial. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *6*(3), 282–308.

O'reilly, K. (2004). *Ethnographic methods*. Routledge.

Onyeka, E. (2013). A process framework for managing implicit requirements using analogy-based reasoning: Doctoral consortium paper. *IEEE 7th International Conference on Research Challenges in Information Science (RCIS)*, 1–5. IEEE.

Parnas, D. L., Asmis, G. J. K., & Madey, J. (1991). Assessment of safety-critical software in nuclear power plants. *Nuclear Safety*, *32*(2), 189–198.

Phosaard, S., & Wiriyapinit, M. (2011). Knowledge management via Facebook: building a framework for knowledge management on a social network by aligning business, IT and knowledge management. *Proceedings of the World Congress on Engineering*, *3*, 1–6.

Pohl, K. (1996). *Process-centered requirements engineering*. John Wiley & Sons, Inc.

Polanyi, M. (1967). The tacit dimension. Anchor. *Garden City, NY*.

Popadiuk, S., & Choo, C. W. (2006). Innovation and knowledge creation: How are these concepts related? *International Journal of Information Management*, *26*(4), 302–312.

Pressman, R. S. (2005). *Software engineering: a practitioner's approach*. Palgrave Macmillan.

Project Management Institute. (2018). *Pulse of the Profession 2018*. Retrieved from https://www.pmi.org/learning/thought-leadership/pulse/pulse-of-the-profession-2018

Royce, W. W. (1987). Managing the development of large software systems: concepts and techniques. *Proceedings of the 9th International Conference on Software Engineering*, 328–338. IEEE Computer Society Press.

Saint-Onge, H. (1996). Tacit knowledge the key to the strategic alignment of intellectual capital. *Planning Review*, *24*(2), 10–16.

Schulze, A., & Hoegl, M. (2008). Organizational knowledge creation and the generation of new product ideas: A behavioral approach. *Research Policy*, *37*(10), 1742–1750.

Seidler-de Alwis, R., & Hartmann, E. (2008). The use of tacit knowledge within innovative companies: knowledge management in innovative enterprises. *Journal of Knowledge Management*, *12*(1), 133–147.

Sommerville, I. (2011). *Software engineering*. Addison-Wesley/Pearson.

Sutherland, J. (2004). Agile development: Lessons learned from the first scrum. *Cutter Agile Project Management Advisory Service: Executive Update*, *5*(20), 1–4.

Viller, S., & Sommerville, I. (2000). Ethnographically informed analysis for software engineers. *International Journal of Human-Computer Studies*, *53*(1), 169–196.

Von Krogh, G., Ichijo, K., & Nonaka, I. (2000). *Enabling knowledge creation: How to unlock the mystery of tacit knowledge and release the power of innovation*. Oxford University Press on Demand.

Wan, J., Zhang, H., Wan, D., & Huang, D. (2010). Research on knowledge creation in software requirement development. *Journal of Software Engineering and Applications*, *3*(05), 487.

Wright, J. D. (2015). *International Encyclopedia of the Social & Behavioral Sciences*. Retrieved from https://books.google.co.jp/books?id=TQaFBAAAQBAJ

Zave, P. (1997). Classification of research efforts in requirements engineering. *ACM Computing Surveys (CSUR)*, *29*(4), 315–321.

List of publications

The result of this research has been published and presented at the 14th International Conference on Knowledge, Information and Creativity Support Systems, 21-23 November 2019, Ho Chi Minh city, Vietnam as follows:

Le Anh Hoang, Naoshi Uchihira (2019). Requirement engineering in knowledge-intensive manufacturing shop floors – a knowledge-based approach. *The 14th International Conference on Knowledge, Information and Creativity Support Systems (KICSS 2019). Accepted.*