The Study on Knowledge Transfer of Software Project Requirements

Xiaohong Shan
The College of Economics and Management
Beijing University of Technology
Beijing, China

GuoRui Jiang
The College of Economics and Management
Beijing University of Technology
Beijing, China

Tiyun Huang
The College of Economics and Management
Beijing University of Technology
Beijing, China

***Abstract*—The effective transfer of software requirement knowledge is one of the key factors of decreasing the requirements volatility, solving conflicts between requirements, ensuring the quality of software project and customer satisfaction. From the point of knowledge management, this paper first classifies the software project requirement knowledge into 4 categories according to the knowledge sources from which software project development team acquires requirements. Then constructs the knowledge transferring model of requirement development and requirement management in software projects, and analyzes the major influencing factors during the requirement knowledge transferring process. At last the strategies that can promote the requirement knowledge transfer effectively are proposed.**

Keywords- software requirement; requirement development; requirement management; knowledge transfer; strategy

I. INTRODUCTION

As shown in many researches, the problem of requirements is one of important causes in software projects' failure [1-9]. The investigation results of IT projects by Standish Group in 1995 suggested, in failed software projects, there are five closely related to the requirements among the first ten failure causes [10]. It is easy to see that if the software project leaps into the system design stage without grasping the customers' requirements, and forming the high-quality software requirements specification that confirmed by the key project stakeholders of customers at the requirement analysis stage, it will inevitably cause the frequent changes during project implementation, and affect the cost, quality, schedule and customers' satisfaction degree of software projects or products directly. So the requirements are crucial to the success of the software projects. In order to increase the success rate of software projects, many studies on software project requirements focused on the requirement volatility [11-12], the software requirement modeling [13-14] etc. These methods provide guidance effect on helping project team acquiring exact requirements, analyzing the requirements rationally, and controlling requirement changes effectively etc.

Recently the development of knowledge management theories and technologies provided a new angle for the study of software requirement management. Na Yisha constructed knowledge-transferring model of requirement development and requirement management in his doctoral dissertation [15]; Li Xiaoming etc gave the effective software requirement management strategies based on the theory of knowledge dissipative structure [16]. From above we can see software requirement, as a knowledge aggregate, its development and management process can be managed from the perspective of knowledge, but there are little further studies on this subject.

Different from past requirement knowledge classifications, this article classifies the requirement knowledge according to knowledge sources, and constructs the knowledge transferring model of requirement development and management, then analyzes factors affecting the requirement knowledge transfer and provides the strategies that can promote the knowledge transfer to decrease the incidence rate of requirement changes, and control the requirement risks effectively.

II. THE CLASSIFICATION OF SOFTWARE PROJECT REQUIREMENT KNOWLEDGE

From the point of knowledge transfer, the knowledge that software project requirement management involves can be divided into four categories according to the knowledge sources from which software requirements is acquired.

(1) Domain requirement knowledge:

Domain requirement knowledge can be divided into vertical and horizontal domain requirement knowledge. Vertical domain requirement knowledge refers to the requirement knowledge particular to some industry. Each industry has some special characters. Software requirement knowledge can't be separated from the characters of the industry; while horizontal domain requirement knowledge refers to the requirement knowledge of mutual sub domains of various industries. The domain requirement knowledge is usually acquired from the domain experts, who are very familiar with the system requirements, design and implementation of the software, the use of software, hardware and technology etc of this domain.

(2) Customers' Requirement Knowledge

Customers are the main source that software team acquires requirements. Customers need to provide what they want to solve. Customers' requirement knowledge is the basis of software requirement analysis and modeling, which determines the functions of software systems.

(3) End users' requirement knowledge

978-1-4244-5316-0/10/$26.00 ©2010 IEEE

End users are the individuals or groups who will use the software to be developed, whose knowledge about software requirements is very important to the function of software systems too. End users and customers can be the same or different individual or groups.

(4) Constraints requirement knowledge

Constraints requirement knowledge is acquired from the stakeholders of software projects, which includes many aspects, such as the constraints of system environment, the budget, schedule and quality that development need to meet etc.

One of the major tasks of software requirement engineering is to accomplish the communication and transformation between requirement-related persons and software development team. Software project requirement management is a process of knowledge management. It manages a series of problems occurred during the process, which takes the requirement knowledge transferring from the stakeholders to the software project team as the starting point, the internal knowledge transferring among project team as the process, and the new knowledge (software products and relevant documents) delivering to the customers as the end point. It experiences knowledge acquisition, transformation, processing, and innovation during the requirement management process.

III. KNOWLEDGE TRANSFERRING MODEL IN SOFTWARE PROJECT REQUIREMENT MANAGEMENT

The knowledge transfer in software project requirement management can be divided into the knowledge transfer of requirement development and requirement management, according to the composition of requirement engineering.

A. The Knowledge Transferingr Model of Requirement Development

Requirement development process includes requirements elicitation, requirements analysis, requirements description, and requirements validation. The knowledge senders includes end users, customers, domain experts and other project stakeholders during the process of requirement development, who send the above four requirement knowledge to the knowledge accepters (namely software project development team) respectively. The knowledge first experience the transformation from the individual knowledge to the organizational knowledge in their respective organizations of knowledge senders, that is to transfer the individual's scattered implicit knowledge to the knowledge base of the organization, in which all the knowledge are shared by all individuals. Requirement knowledge has both explicit and implicit knowledge. In order to transfer the knowledge to the development team accurately, it is necessary not only to express explicit knowledge exactly, but also promote the effective transferring of implicit knowledge. There are two things needed to do. One is knowledge socialization by knowledge senders, that is the knowledge exchange between individuals of each knowledge sender, which will form organizational shared knowledge finally; the other is knowledge externalization through appropriate ways by requirement analyst, that is to transform the implicit knowledge of knowledge senders into explicit knowledge, which can be transferred to development team. After acquiring the knowledge from senders, requirement analysts need to analyze the knowledge, and make negotiation about unnecessary, incomplete, infeasible, and conflict requirements, until the consensus of the final requirements are reached. It is an iterative process. Consistent requirements can be described by proper ways, usually requirement specification, which should be sent back to knowledge senders for validation to ensure the correctness. If senders raise an objection in the validation results, the new process will restart, otherwise requirement can be transferred internally among the development teams with the form of requirement specification, use-case documents, and analysis models etc. The part of coarse dashed line frame describes the knowledge transfer process of requirement development in Fig 1.

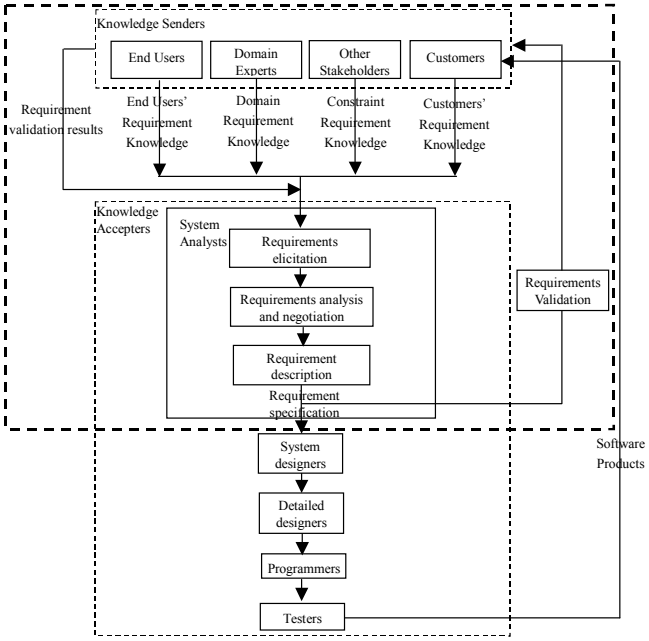

Figure 1. The knowledge transfer of requirement development

B. The Knowledge Transferring Model of Requirement Management

The results of requirement development such as requirement specification can be taken as the requirement baseline after formal review and approval. In the following development process the requirement management is needed, which manages changes of requirements.

When the requirement change is sent to requirement analysts, they will analyze the change and its effects, thus decide whether to accept it and give a feedback of the results and their reasons to knowledge senders. If to accept, analysts will modify the specification, and send them to designers, programmers and testers in order to transfer the changed requirement knowledge to other persons of the development team, thus to complete the internal changed requirement knowledge transfer process. The knowledge transferring process of requirement management is shown in Fig 2.

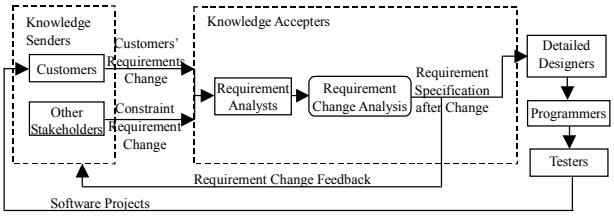

Figure 2. The knowledge transfer of requirement management

IV. THE ANALYSIS OF FACTORS AFFECTING THE EFFECT OF SOFTWARE PROJECT REQUIREMENT KNOWLEDGE TRANSFER

To improve the requirement knowledge transfer process it is necessary to analyze the influencing factors of knowledge transfer.

A. The characteristics of software requirement knowledge itself

Software requirement knowledge has the characteristics of uncertainty, volatility, subjectivity and complexity; it can't be acquired and determined accurately and easily. It has not only explicit knowledge that can be acquired by methods of interview, questionnaire, use-case and scenario, but also plenty of highly individual knowledge, which are the results of individual's long-term accumulation, learning and creation, and are difficult to express, transfer and intimate. Knowledge accepters need more time and energy to make the implicit knowledge explicit, thus to acquire, learn and determine the requirement knowledge. The higher the uncertainty and subjectivity of software requirement knowledge, the more difficult the requirement knowledge transfers.

B. Software requirement knowledge senders

From the view of the subjective willingness of knowledge senders, customers' and other stakeholders' willingness of sending knowledge is strong, but end users are more reluctant to transfer knowledge because they worry about the effects the software may bring to their future job. Domain experts do not have strong willingness because they are not closely related with the software development project; From the view of the knowledge the senders own, the implicit degree of end users', customers' and domain experts' knowledge is relatively higher and more difficult to transfer, while other stakeholders, whose requirements are the constraint knowledge, are lower in implicit degree and thus easy to transfer; From the view of confidence degree that knowledge senders have on knowledge accepters, the more customers' trust about the ability of development team, the more confidence about the future benefits the software products will bring, the better the effects of knowledge transfer.

C. Software requirement knowledge accepters

From the view of the subjective willingness, the willingness of software requirement knowledge accepters is generally strong because requirements determine whether the system can satisfy the needs of customers and whether the development is successful directly. The knowledge that the accepters own can be considered from two aspects. On one hand, as accepters of external requirement knowledge from customers, system analysts' knowledge is generally major knowledge of computer and the experiences of past development, and it is very difficult for them to study and assimilate the requirement knowledge of customers. On the other hand, as accepters of knowledge from system analysts, designers, programmers and testers' knowledge of computer make them easy to understand and learn the knowledge system analysts transfer; From the view of the ability of maintaining knowledge, normative documents can be beneficial to the transformation and maintaining of requirement knowledge, and ensure the transformation of requirement knowledge in the time-dimension.

D. The channels of software requirement knowledge transfer

The channels of software requirement knowledge transfer involve two aspects. one is the requirement knowledge that can be expressed clearly by linguistic signs. They can take documents, ontology, and UML etc as their transferring channels. The other is knowledge hidden in the individuals, which is hard to express by languages and symbols. They can be made explicit by investigations, interviews and informal channels such as dinner party, thus archiving the goal of acquiring requirement knowledge. The thing is, it is not certain whether these channels can ensure making the implicit knowledge explicit correctly and transfer to the knowledge accepters effectively. It is affected by many factors. This is also the last influencing factor we will mention in the next part.

E. The distance between the software requirement knowledge senders and accepters

The factor of distance indicates the gaps between the knowledge senders and accepters. The distance between the software requirement knowledge senders and accepters include knowledge distance, physical distance, and cultural distance.

(1) Knowledge distance

Knowledge distance is divided into two aspects. One is the knowledge distance between knowledge senders and system analyst. The knowledge that knowledge senders own are the professional knowledge of their own domain or enterprises, and that of system analysts are knowledge of software engineering and computer, as well as experiences. So their knowledge distance is relatively larger, and it is relatively difficult to transfer their knowledge. If system analysts own rich experiences, especially they have developed the system of similar domain or enterprises, the overlapping knowledge can be increased, and their knowledge distance will be shortened, thus knowledge transfer is improved. The other is the knowledge distance between system analysts and designers, programmers, testers. Because they usually have the same or the similar background of professional knowledge, what's more many enterprises usually make regular job rotation to shorten the knowledge distance and promote the effective communication and knowledge sharing among personnel. What we should not ignore is excessive overlapping of knowledge can hinder the knowledge innovation. To form the knowledge gap systematically and to combine persons who have difference knowledge and experience together is one of necessary condition of knowledge transfer and creation.

(2) Physical distance

The influences of physical distance on the effects of knowledge transfer become increasingly weak. As long as the cost allowed, physical distance's influence can be made up by many methods. Either videoconference or prototyping can bring the similar effect as face-to-face communication.

(3) Cultural distance

The cultural differences between software requirement knowledge senders and accepters also affect the requirement knowledge transfer. The shorter the cultural distance, the easier the knowledge transfers. Software development teams have the characteristics of young personnel, less evident hierarchy system, and relaxed atmosphere compared with general enterprises. What software development team faced with are the customers in every field. The corporate cultures are different from each other. For example, the corporate culture differences between the traditional state-owned enterprises and software enterprises are relatively bigger, it is relative difficult to transfer requirement between them; while the corporate culture differences between the telecom enterprises, high-tech enterprises and software enterprises are relatively smaller, and it is relatively easy to transfer their knowledge.

V. THE STRATEGIES PROMOTING REQUIREMENT KNOWLEDGE TRANSFER EFFECTIVELY

In order to promote requirement knowledge transfer correctly and effectively, we put forward the following strategies based on the analysis of factors Section IV described.

(1) To establish the knowledge transfer platform

Effective ways of knowledge transfer can shorten the distances between software requirement knowledge senders and accepters, reduce the loss of the process of knowledge transfer, and increase the effectiveness and efficiency of knowledge transfer. For example, UML and ontology are both good knowledge transfer ways.

(2) To construct the reusable requirement knowledge base

The reusable requirements include the fellows' experiences about requirement elicitation and management, requirement specification and domain models. Constructing the component base of requirement knowledge can help us to utilize the requirement knowledge of successful projects effectively. More importantly, acquiring the knowledge of usable requirement knowledge base realize an effective knowledge transferring process, and lay the foundation for knowledge sharing.

(3) To make prototype

The software prototype, which makes the requirement more authentic and vivid, decreases the understanding differences of requirement knowledge senders and accepters. A prototype in front of users can activate the thoughts of users, promote users' explanations about requirement, and thus get faster feedback. It can be beneficial to getting consensus between knowledge senders and accepters as soon as possible, and reduce the risks of excessive requirement change and users' dissatisfaction.

VI. CONCLUSION

This article classifies the software requirement knowledge from the view of the sources from which software development team acquires requirement knowledge. Based on the classification the knowledge transfer models of requirement development and management are constructed. These models highlight the particularity of software requirements compared with past knowledge transfer models from the view of the knowledge's own characteristics. Furthermore, this article presents an in depth analysis on main influencing factors during the process requirement knowledge transfer, and proposes the strategies that can promote the effective requirement knowledge transfer.

REFERENCES

[1] Barry W Boehm. Software risk management: principles and practices. IEEE Software, January 1991: pp.32-41.

[2] Chuk Yau. A Quantitative Methodology for Software Risk Control. IEEE 1994: pp.2015-2020.

[3] Mark Keil, Paul E. Cule, Kalle Lyytinen, Roy C. Schmidt. A Framework for Identifying Software Project Risk. Communications of the ACM, 1998, 41(11): pp.76-83.

[4] Liu Renhui, Zhai Fengyong. Multi-Attribute Evaluation for Software Project Risk[A]. International Conference on Wireless Communications, Networking and Mobile Computing, WiCom 2007. Shanghai: Inst.. and of Elec Elec. Eng. Computer Society, 2007.

[5] Wen-Ming Han, Sun-Jen Huang. An empirical analysis of risk components and performance on software projects. Journal of Systems and Software, 2007, 80(1): pp.42-50.

[6] Dey, Prasanta Kumar, Kinch, Jason, Ogunlana, Stephen O. Managing risk in software development projects: a case study. Industrial Management & Data Systems, 2007, 107(2): pp.284-303.

[7] Barki H, Richard S, Talbot J. Toward an assessment of software development risk [J]. Journals of Management Information Systems, 1993, 10: pp.203-223.

[8] Liang Tao, Ou Lixiong, Huang Kexin. Cluster Analysis Based Software Project Risk Trends Study. Journal of Information Engineering University, 2006,1: pp.88-90, 102.

[9] Dan X Houston, Gerald T Mackulak, James S Collofello. Stochastic simulation of risk factor potential effects for software development risk management. Journal of Systems and Software, 2001, 59(3): pp.247-257.

[10] Standish Group. Chaos 1995. http://www.standishgroup.com, 1995.

[11] Talha Javed, Manzil-e-Maqsood, Qaiser S. Durrani. A Study to Investigate the Impact of Requirements Instability on Software Defects. ACM Software Engineering Notes, 2004, 29(4): pp.1-7.

[12] G.P. Kulk, C. Verhoef. Quantifying requirements volatility effects. Science of Computer Programming, 2008, 72: pp.136–175.

[13] Shao Kun, Liu Zongtian, Hu Xuegang, Li Xinke. AML: Oriented-Requirement Multi-Agent System Modeling Language. PR&AI, 2007, 20(1): pp.131-137(in Chinese).

[14] Li Furong, Zhang Yang, Zhang Jingjun, Wang Meng. Aspect-Oriented Requirements Modeling Based on UML. Computer Applications and Software, 2007, 24(6): pp.35-36,81(in Chinese).

[15] Na Yisha. Research on Knowledge Transferring Model and Strategies for Requirement Engineering. Tianjin University, 2006: 1-125(in Chinese).

[16] Li Xiaoming, Sun Linyan, Wang Yingluo. Research on Software Requirement Management based on Knowledge Management. R&D Management, 2005, 17(2): pp.28-32,39(in Chinese).