Research on Logistics Software Design and Development Process Bases on Organizational Learning

Zhongwei Wang，Si Qu
School of Logistics
Central South University of Forestry and Technology
Changsha, 410004, China
E-mail: wangpmp@163.com

***Abstract*—This paper describes a cycle in which organizational learning and software development serve each other, presents the concept of organizational learning cycle method. Experience in a real software development project has led us to identify several key challenges for software design and development through organizational learning. A discussion of these challenges presents steps taken toward addressing them, as well as barriers remaining.**

Keywords-Organizational learning, Group memory, Systems design, Collaborative work

I. Introduction

Many organizations view learning as a stand-alone activity separate from daily work. Workers are trained away from their workplace, and then are assumed to perform their jobs according to the procedures learned during training. We argue for an integrated view of organizational learning, in which workers learn as they do work, and the organization learns from the experiences of workers [1].

In the context of software development, organizational learning focuses on recording knowledge gained through experience, and subsequently making that knowledge available to other developers when it is relevant to their particular task. A central component of organizational learning is a repository for storing knowledge – a group memory. However, the mere presence of a group memory system does not ensure that an organization will learn. For sustained organizational learning, two seemingly disparate goals must be served simultaneously:

1) Group memories must serve work by making stored information relevant to the task at hand, and
2) Group memories must be extended and updated as they are used to support work practices.

This paper considers organizational learning in the context of logistics software design and development. An organizational learning approach to software development uses an organization's accumulated knowledge of the development process and application domains as the basis for design. This perspective of software design highlights the relationship between the knowledge needs and products of a specific project, and the accumulated knowledge of past projects that are stored in a group memory. When a development group in the same domain for a while, organizational learning from past experiences becomes intriguing: many questions arise repeatedly, previous projects may have uncovered knowledge that is applicable to several projects, and newcomers who may have forgotten must be familiarized with the domain and design knowledge.

The next section describes a real process of software design project within an organization. This project illustrates key challenges for organizational learning in the context of logistics software design. The following section presents an organizational learning cycle as our integrated approach for meeting these challenges. It is a warehouse and distribution software development project oriented to the distribution center of Hunan province highway service area. In the whole design and development process, we are confronted with many difficulties, especially in the requirements analysis phrase. In the following part, we will demonstrate how we solve the problems through organization learning method.

II. Challenges for Logistics Software Design through Organizational Learning

Combined with a logistics software development project, this section illustrates key challenges for organizational learning in the context of logistics software design. While working with an software development group, we became involved in a problematic development project. Prior to our involvement, the user group, i.e. the logistics distribution center had contracted with an outside development contractor to produce a new management system. The development strategy followed a classic contract development model in which the user group first produced a request for proposals that described what they wanted in their new system. The development company then sent some people to talk to representatives of the user group. The representatives of the user and development groups negotiated the system requirements and produced a requirements specification. The purpose of the specifications document was to “freeze” the requirements, and to serve as a contract that described the responsibilities of the development group. The contractors then assigned programmers to the project and began to design and implement a system based on the specifications document.

At this point we became involved in the project. Together with the distribution center, we decided the biggest problem the project suffered was the lack of interaction between the user group and software designers. This lack of interaction had two dimensions.

- First, the distribution center did not feel that the designers understood the problems of the application domain. The users felt their knowledge was relevant to the project and that the specifications document did not reflect this knowledge.
- Second, the development process did not provide for ongoing interaction between distribution center and software designer. When the project encountered difficulties, the obvious solution of discussing the problems was not available because the project was set up as though the specification document would be a static contract between designers and users.

To address these problems, we adopted an evolutionary development approach. The goals were to activate the domain knowledge of the users-the worker in the distribution center and, to keep them involved throughout the project. To achieve these goals I opened up the process to users and made communication the driving force for software design by using diagrams and prototypes and ground discussions. During discussions designers took notes, which were then added to a group memory.

However, two new challenges were encountered. First, we had difficulty in capturing activated knowledge that was produced in the course of communicating about the design, even after we had assigned the exclusive role of scribe to one designer. Second, we could see that success in capturing all the knowledge produced by our participatory and evolutionary approach would result in a huge group memory. Making use of this memory later in the project or in a subsequent project would require that the stored information was made relevant to the design task at hand.

In summary, the key challenges identified for informing software design through organizational learning are:

1) Activating relevant knowledge,
2) Keeping the distribution center involved and interested,
3) Capturing activated knowledge,
4) Making stored information relevant to the design task at hand. The next section discusses these challenges and how they can be integrated into an organizational learning cycle.

III. Toward an Integrated Approach: An Organizational Learning Cycle

Our approach to informing software design through organizational learning is to realize a continuous cycle in which individual projects serve group memory while group memory serves individual projects. Projects serve the group memory by adding new knowledge that is produced in the course of doing design, such as innovative design products and practices, rationale, and communication. Group memory serves projects by providing relevant knowledge when it is needed, such as solutions to similar problems, design principles, or advice.

The organizational learning cycle involves three of the four challenges (activating knowledge, capturing activated knowledge, and making stored information relevant to the task at hand), which form a feedback cycle of interdependent activities. Breaking the cycle at any given point will have an adverse impact on all of the activities [2].

The remaining challenge is to keep the distribution involved and interested in the development process, which is the driving force for the organizational learning cycle. Keeping the distribution center involved drives the cycle by continuously activating new knowledge, which can then be captured and used to activate more knowledge. The components of the organizational learning cycle are now described in more detail.

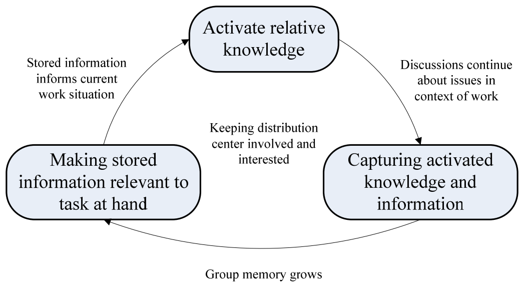
Activate relative knowledge
Stored information informs current work situation
Discussions continue about issues in context of work
Keeping distribution center involved and interested
Making stored information relevant to task at hand
Capturing activated knowledge and information
Group memory grows

Figure 1 An organizational learning cycle

A. *Keeping distribution center involved and interested.*

Interaction between worker in distribution center and software designers is the driving force of the organizational learning cycle, making it an evolutionary approach. Contract-based approaches are not conducive to organizational learning because they fail to exploit the learning opportunities that software design presents. These approaches can push projects to write elaborate specifications of poorly understood user interfaces, before the problems to be addressed by the systems are well understood. Once the specification is frozen, it is assumed that no further interaction between designers and users need take place. By failing to keep users involved and interested throughout the development process, these approaches may cut off communication before design issues are understood. Besides leading to design errors, cutting off communication is also detrimental to organizational learning because knowledge potentially useful to the organization is never activated.

Software development is an opportunity for organizational learning if those with the relevant knowledge are kept continuously involved. There is growing evidence that system requirements are not so much analytically specified as they are collaboratively evolved through an iterative process of consultation between distribution center and software designers. This interactive process is necessary because distribution center can initially know their requirements only vaguely at best. New requirements emerge during development because they cannot be identified until portions of the system have been designed or implemented [3]. It is the method to improve and perfect the final result by combining the distribution center and software developer and designer in the whole course.

B. *Activating relevant knowledge.*

A prevalent problem facing software design projects is the so-called thin spread of application knowledge, in which software designers have too little knowledge about the application domain to make informed decisions. A major factor contributing to this problem is that much application domain knowledge is tacitly understood by domain workers (for example, the process of commodity distribution is well known by the workers in distribution center). This tacit knowledge is essentially taken for granted because of its familiarity, or can be difficult to articulate.

Activating knowledge means to make knowledge explicit so it can be shared by others. Many methods can achieve the goal, such as prototypes, which can activate tacit knowledge by creating breakdowns in tacit understandings. Breakdowns are caused when prototypes fail to act as expected, making distribution center more aware of what they formerly took for granted. This activated knowledge can help both distribution center and software designers to better understand the application domain and what the new distribution center management system should be like [4].

In the phase, we found informal diagrams and prototypes to be effective for engaging distribution center, service areas and developer in constructive discussions. We also found that the knowledge these discussions activated could be difficult to capture. This difficulty leads to the third challenge for organizational learning in software design: capturing activated knowledge.

C. *Capturing activated knowledge.*

If software development is to contribute to organizational learning, activated knowledge must be captured and stored in a group memory. Feeding information into a group memory for later use by others is a long-term investment in the future of the organization. However, our experiences suggest that an integrated approach, intertwining design work with the collection of information, can decrease the perceived cost of contributing to a group memory [5]. In the second phase of the project this cost was negligible, since the design products were created for the immediate needs of the project and then simply collected in the memory. The contents of the group memory in the second phase of the development project indeed grew quite rapidly. However, the contents of group memory were all contributed by designers. Thus, while we succeeded in lowering the perceived cost of contributing to group memory, a bottleneck formed at the point where information representing the distribution center's comments was entered into the group memory. A remaining challenge to capturing activated knowledge is to move from seeing the group memory as a holder of organizational knowledge [6] to seeing the group memory as a medium of communication through which distribution center and software designers speak for themselves.

D. *Making stored information relevant to the design task at hand.*

To inform software design, group memories must provide distribution center and software developer with the information they need when it is needed. Group memories are not useful simply because they store a great deal of information. They are useful when they make stored information relevant to the task at hand. The information needs of development projects cannot be completely anticipated in advance. Instead, information needs occur in the context of trying to accomplish a particular task. So at this time, distribution center, every service areas and software designer should strengthen cooperation, interaction on the basis of group memory to extract useful relative information for the information system development.

IV. CONCLUSIONS

This paper has argued that organizational learning should be an integral part of practical work. The integration of organizational learning and practical work is examined in the context of software design. An approach for informing software design through organizational learning was presented, and the following key challenges were identified:

1) Activating relevant knowledge,
2) Keeping users involved and interested,
3) Capturing activated knowledge, and
4) Making stored information relevant to the design task at hand.

Through combined with the real case we demonstrate the organizational learning cycle method to solve the problems of interaction between user group and software developing group existing in software design and development phrase.

REFERENCES

[1] Hailey, John and Rick James. Learning leaders: the key to learning organizations[J].Development practice 12(3/4):398-408.

[2] Chen guoquan, Ma meng. Organization learning-present situation and outlook[J].Chinese Journal of management science,2000(1).

[3] Gupta A, Thomas g. Organizational learning in a high-tech environment: from theory and practice[J]. Industrial management and data system,2001(8).

[4] Sun xiaoqiang.The role of organizational learning in knowledge management process in the view of organizational learning process model[J]. Technoeconomics and management research.2007(1).

[5] Chen yuanyuan. The couple research on organization learning, knowledge management and organizational innovation[J].Library and information service.2010,54(2).

[6] Binney, Derek. The knowledge management spectrum-understanding the KM landscape[J]. Journal of knowledge management.2005,5(1):33-42.