Organizational Knowledge in a Software Company Growth with Awareness of a Three-Stage Evolution Model

Toshihiko Yamakami[†‡]
[†]Research and Development, ACCESS
[‡]Graduate School of Engineering, Kagawa University
Toshihiko.Yamakami@access-company.com

Abstract

Sustained challenges in organizational knowledge support include how to evolve according to the problem solving domain dynamism. For example, while a small company grows, the knowledge support requirements shift. With the emerging globalization, it is an every day challenge for organizational knowledge support. In this paper, the author made a case study on a growing software vendor company to identify the impact of the organizational dynamism. From observation, the author proposes 3-stage models in both of vertical and horizontal evolution dimensions. The author describes the practical guidelines for knowledge support systems from a perspective of dynamism.

Keywords: Knowledge Evolution, Corporate Dynamism, Company Growth

1 Introduction

The challenges in creative problem solving include the dynamic adaptation to the problem frameworks. The dynamism in a software development company is analyzed with implications to the multiple levels of problem domains. During the corporate business growth, there could be a significant impact on problem solving frameworks. With globalization, it is a vital challenge to cope with this growing dynamism in the organizational knowledge support systems. The companies today encounter every day challenge to cope with the increased complexity and globalization. When the author joined the company, it was a 90-employee 1.1 billion-yen-sales company. Now it grows to be a 1500-employee 30 billion-yen-sales company. The revenue growth is depicted in Figure 1. The employee growth is outlined in Figure 2. This growth and accompanied evolution give challenges to the organizational knowledge sharing structure.

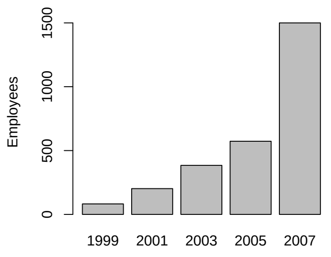

Figure 1. Headcounts of the Corporate-group

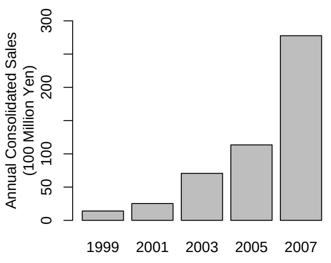

Figure 2. Revenue growth (unit: 0.1 billion yen)

In this paper, a case study with a growing software vendor is discussed with the three different levels: software development level, project management level, global corporate management level. The framework changes during the growth are outlined to identify the today's corporate demands on the creative problem solving and organizational knowledge support.

The author performed a case study to capture the typical knowledge of a software vendor company in Japan. During its growth, it was recognized that the organizational knowledge shifted from program knowledge, project knowledge and overall business structure. Software vendors are unique where each company converts human intelligent activities into software, the final product. This uniqueness brings a challenge to software vendor to focus its competence with intelligent activity support and knowledge sharing. In this paper, with a case study with a growing software vendor organizational knowledge shifts, the author gives some considerations to the organizational knowledge dynamism from time-dimension dynamism with corporate growth. The author summarizes some guidelines for organizational knowledge sharing in the dynamic situation encountered in the today's competitive business environment.

2 Research Purpose

The organizational knowledge is impacted by the industrial conditions and competences of each company. The dynamism on the corporate external conditions is a common challenge for corporate activities, especially under the impact of globalization. The aim of this research is to identify the frameworks and guidelines for organizational knowledge sharing under dynamism.

3 Related Works

The dynamism of organizations is non-negligible part of organizational knowledge. There are multiple approached how to capture the dynamic part.

For example, the past organizational knowledge systems like Answer Garden [1] had an assumption that the organizational knowledge did not have a rigid structure. The goal of such a system was to achieve a relatively improved status with incremental knowledge acquisition. Know-how sharing had similar literature[2].

SECI model [3] tried to capture the dynamism between explicit knowledge and tacit knowledge. It indicated a framework not to miss the large part of tacit knowledge in the knowledge sharing process. Kunifuji indicated importance of positive feedbacks among participants from practical case studies[4].

These approaches tried to capture the dynamism issue from multiple aspects. It is important to identify the organizational process to absorb the qualitative dynamism on corporate growth In globalization, corporate systems need to cast off with environment changes and internal growth. To our best knowledge, there was no literature on codifying how to cope with this challenge.

4 Complexity of Issues

4.1 Complexity in Software Development

Software development is unique in manufacturing that it depends on human resources. It is tightly coupled with knowledge sharing. It is essential part of software development to cope with problem domain complexity and to convert to the complexity into machine-processable forms. Any modularized software development is not free from this complexity origin in the real world. This conversion to the machine-processable forms is resolved with architecture, library, or other development methodologies.

Another source of complexity of software problem solving is project management. When a software product increases its size, complexity management becomes a crucial part of business. Large-scale software goes beyond a small development team. For example, a browser client in a handset reaches a million lines of code. The number of testing items exceeds 100 thousand. Including other software in a handset, a total development size of client software reaches multiple millions lines of code and more-than-a-million testing items. The division of software process into requirement identification, coding and testing requires intensive coordination.

Current globalization increases offshore development like China and India in Asia, Eastern European countries in Europe. It becomes a critical factor to manage global resources in software development. When a company grows beyond country boundaries with expanded marketing and development networks, even the inhouse development has complexity comparable to the past inter-organization development. New development paradigms like open source add further complexity.

Software manufacturing depends on human resources. It is a manufacturing process with human resources from requirement to release process. It is important to raise awareness in sympathy, we-ness, and motivation in order to control global software development process. Otherwise, it is difficult to improve software quality in the global development scenes. The globalization not just impacts the software development flow, but the software business models impact development process [5]. The business model issues also influence the complexity of organizational knowledge support.

4.2 Information Appliance Software Development

The client software in a mobile handset includes browser, mailer, Java, Flash and so on. The browser software reaches a million line of codes. Other software increases size and complexity. The most complicated software in a PC client was Windows XP, estimated software size was 5 millions of code. The Windows Vista may be further complicated, however, the client software in a mobile handset is not a small one today. It is to be noted that a 100g handset with a limited display requires this size of complexity. The size of software is comparable to ones in the financial systems in 1980's.

There are several characteristics in the mobile handset software that are common to information appliance software. They are as follows:

- Diversity of environments and capabilities
- Parallelism due to the short product life cycle

The underlying execution environment of software in a mobile handsets is diverse. REX (a binary environment is BREW) in CDMA-1x. Symbian and Linux are used in GSM-based systems. The emerging iPhone uses OS X. Samsung press releases a Window-XP-based mobile handset. This diversity increases code size and complexity of porting process. The mobile handsets have biannual product releases, which means twice a year. This increases development complexity. The product process has a parallelism with the bug fixing from multiple previous releases, development for the current and the future releases, testing, performance improvement, quality improvement for core software. The parallelism includes product versions, libraries, and core versions. These construct multiple parallel source code sets. For each product, the simulator version, debugging versions, real machine version produce multiple parallel binary versions. For real machine version, it may use different library codes to reduce the final foot print size. This environment leads to the integrated management on multiple source code sets and binary versions.

These diversity and parallelism lead to the increased complexity in development software. It adds complexity on the inherited one of software.

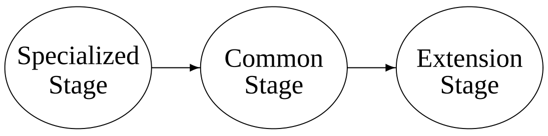

Figure 3. 3-Staged Model of Software Architecture

4.3 3-Staged model of software architecture

A typical software architecture evolution pattern is illustrated in Figure 3. The initial stage needs a particular competence in a limited domain. The initial success of software comes from some particular platform, a particular problem domain, or a particular customer. Next, the software architecture moves forward to the common stage. The common architecture enables the solution applicable to a wider range of applications or problem domains. Smaller and modular parts enable the simplified manufacturing process and minimized maintenance. When software succeeds in this stage, the software moves forward to the extension stage, where the company adds value-added enhancement and customization to wider problem domains with keeping the common core accomplished in the previous stage.

It is interesting that the browser software in mobile handsets follow this evolution path.

The browser software started from the focus on the compact code size. The mobile handset CPU was power-less and bundled memory was small. The requirements from the wireless carrier were to maintain the four key factors (a) size, (b) weight, (c) cost and (d) battery life as well as enabling Internet access. This priority was a little bit different from the common software engineering. This featured the initial stage characteristics.

When software reached the common stage, the micro-browser became a critical part of mobile handset software. In this stage, the priority of development shifted toward (a) software quality, (b) reliability, (c) ease of porting, and (d) short market lead-time. The size of object code was not in the top priority list. The new created common core codes increased the total size of execution code. The binary size increased 5 times, and the common core codes were established. The portability was improved. This featured the common stage characteristics.

Next, the software entered into the expansion stage. The common core started to produce the micro-browser codes for automotive systems, game consoles and digital TVs as well as mobile handsets. A new framework to cope with multiple profiles was created. Portability was further enhanced to cope with the differences of bundled window systems or with real-time OS facilities. This featured the extension stage characteristics.

This evolution path follows the common software evolution pattern.

5 Organizational Knowledge Sharing Two-dimensional Model

5.1 Two-dimensional model to cope with corporate evolution

Organizational Knowledge with Dynamism needs a dynamic model to codify the organizational knowledge and its evolution process. The author considers the following items for the model requirements:

- Coping with Software development scaling
- Coping with Globalization and its impacts on development process

In order to address these requirements, it is important to consider the qualitative shifts according to the corporate growth. The author proposes a two-dimensional model as outlined in Figure 4.

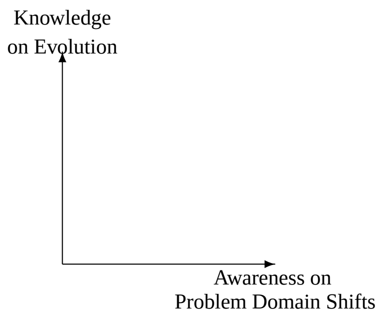

Figure 4. A Two-dimensional Evolution-awareness model

5.2 Three-Staged Model for Evolution

In relation to the software architecture evolution, the author recognized the following three stages in the software vendor growth in development

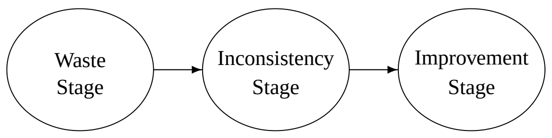

Figure 5. Three-Staged Model of Software Process Know-how

- Centralized control on source codes
- Centralized control on bug lists
- Encouraged sharing on Know-how on library usage
- Encouragement on unified writing rules for headers and functions

Figure 6. Improvement Know-how examples frequently found in the Waste Stage

process and project know-how sharing. They are not one-to-one matching to the software architecture shifts. However, there is some parallel similarity over these three stages. The stages are outlined in Figure 5.

- Waste Stage: Human resources were wasted with lack of know-how sharing
- Inconsistency Stage: Inefficiency was found in the lack of process know-how
- Improvement Stage: No silver-bullet, incremental know-how improves the situation in a slow pace

In the waste stage, it was often found that the software man-month was not efficiently used. The know-how on coding rules, library know-how, know-how on error-prone issues (memory allocation and release, et al) improved the problem solving. A wide range of software know-how was used to improved the problematic projects in this stage. Examples of know-how are depicted in Figure 6.

In the Inconsistency Stage, we can find many pieces of know-how are shared among members though the trouble-shooting in the previous Waste stage. With the increase of size and complexity, we can find an increased number of project management know-how. The project improvement know-how examples are outlined in Figure 7.

- Identifying the bottlenecks in the release process
  - Recognizing diversity in codes, with debugging, without debugging, with simulator, with real machine
  - Reconfirming the efficient parallelism in the task distribution
- Reconfirming the coders with their own testing
  - Identifying the inefficiency of the delayed debugs after coding
- Encouraging the use of static tools like syntax checkers

$\rightarrow$ Minimizing the backward motion in process

Figure 7. Examples of the project improvement know-how found in the Inconsistency Stage

- Checking source revision history
  - Confirming the entropy of code decreases over time
  - Confirming the structured-ness of code is improved over time
- Checking the large-size of source file
  - Checking inefficient copy-and-paste of code fragments
  - Checking the copy-and-paste without understanding functions
- Checking transparent releases

$\rightarrow$ Incremental improvement

Figure 8. Examples of know-how in the Improvement Stage

Many instant problems were solved during the previous two stages. The know-how to cope with the past problems is shared. In this Improvement stage, it is difficult to identify the instant solution to the problematic projects. In this stage, it is typical to accumulate incremental programming know-how and project know-how, especially specific to each project. It is important to identify the bird-eye view of the total project. However, there is no silver bullet in this stage. The know-how examples in the Improvement stage are shown in Figure 8.

In the project management, there are three stages as follows:

- Programmer gaps: Bottom-up programming without clear management

- Functional Inefficiency in Team: Inefficient manager role, missing technical leader

Table 1. Three staged Management Issues

Programmer -bottlenecks	Improvement of programmer skills
Project -bottlenecks	Improvement of debugging/release process
Globalization -bottlenecks	Improvement of Global processes

- Work-split inefficiency: work-split does not work well in scaled-up projects

In the Improvement stage, the corporate management issues become visible more than the case-by-case project issues. Especially, success software tries to cross the country-boundaries, which initiates a trigger to expose the corporate management issues. The obstacles in corporate management are outlined in the three-stage model depicted in Table 1.

5.3 Problem-awareness Dimension

The author became aware that the problem-awareness dimension is important when the issues of globalization became critical. The homogeneity and shared-culture context are lost in the global software development. The lack of this awareness makes the problem more difficult to deal with. Today's global software vendors do their work crossing country-boundaries. Team members around the world have different cultures, economic infrastructures, and we-ness feeling when they are engaged in the global development process. It is inevitable to deal with this issue in the global software development company.

In order to enter this stage, software companies reach the Improvement stage to satisfy the large customer base with sufficient software quality.

The author summarizes the globalization stages in the Figure 9.

The one-way globalization takes place when only one of marketing or manufacturing is global-ized. There are two cases: one is the case where the customer is the domestic and the manufacturing is globalized; the other is the customers are globalized and the manufacturing is centralized. In the two-way globalization, both of marketing and manufacturing are globalized.

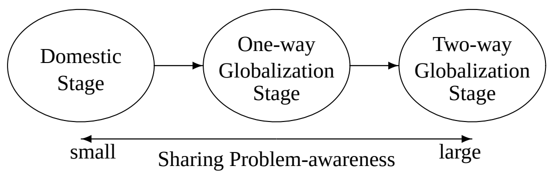
Domestic Stage
One-way Globalization Stage
Two-way Globalization Stage
small
Sharing Problem-awareness
large

Figure 9. 3 Staged View of Software Manufacturing Globalization

5.4 Embedded software engineering specific issues

Information appliances increase their network capabilities. The information appliance software engineering has some common features as the Microsoft software development approaches in the first half of 1990's [6]. For example, the following items have common aspects:

- Every day builds
- Loose specification management different from server side software development
- Software development to deal with changes
- Increased awareness on quality control

It is to be noted that the today's information appliance software development is a combination of the large-scale system development and short turn-around time small-scale system development. This duality brings the following issues:

- The problem solving does not solely depend on technologies
  - Communication Skills
  - Awareness of the tacit situations
    - Dealing with a wide open-ended situation
    - Imagination of the open-ended testing

With some interviews with development facilitators, it is noticed that the good management needs a 6-th sense to detect the good development and poor development in an early stage. The sixth sense is the detection to differentiate the well-controlled ness of the software project. It is triggered by a simple question. When the answer was too simple, or the answer was given too quickly, the 6th sense detects something wrong may happen. This is based on the skills incubated in long experience. It is a challenge to transfer such a sensible detection skill.

The need to facilitate the time-scale problem solving is increased. For example, in the embedded software development, it needs a parallel management on debugging in the past-derived versions and delivering of the new core versions for the future development. Those maintenance and design decision making moves forward in parallel in large-scale in the today's software vendor.

To facilitate the human resource development in remote development, it is important to raise awareness using online meeting. This is based on the importance of the problem-awareness raising. It is still a future issue whether this problem should be solved in a human relation expansion. Sometimes it enables bottom-up skill transfer using human networks. In some cases, this bottom-up knowledge transfer does not work well beyond the cultural boundaries.

6 Guidelines to Knowledge sharing systems

The order-made software development companies can exist in a different scale. Package software development companies are difficult to grow. There are no medium or small size package software companies. There is only one seat for the worldwide packaging software company. In other cases, it is difficult to survive and continue to attract customers. The success packaging software companies are not many. Few examples include Microsoft and Adobe. They are facing with the challenges from SaaS (Software as a Service) business models [7]. This scarcity of examples makes the growing software vendor case studies difficult. From the case study, the author lists guidelines for organizational knowledge in scaling software vendor in Table 2.

It is necessary to facilitate flexible design with imagination to the organizational and process evolution. Organizational knowledge fundamentally tries to improve productivity. Increased productivity brings a company to lead to M&A growth. This illustrates the inherited orientation toward growth in organizational knowledge. Today's dynamic corporate scenes reiterate this as-

Table 2. Guideline for organizational knowledge systems to deal with growth dynamism

Awareness of Growth	Preparedness for growth Quantity-to-quality transfer awareness
Awareness on problem domain shifts	Focus on problem awareness as well as knowledge Stage-awareness
Readiness for globalization	Multi-lingual, translation、Gateway for cross-culture knowledge sharing

pect of organizational knowledge. Conceptually, it is acceptable, however, it is difficult to deploy in the real world organizations.

The telephony and Internet technologies are improving. However, the challenge to identify the corporate knowledge and design to facilitate it in real world scenes are still challenging to researchers and businesspersons.

7 Conclusion

The time-dimensional issue in the software company growth is discussed in the viewpoint of organizational knowledge growth. A case study in an embedded software vendor was performed. Organizational knowledge inherits scale-up of a company. Growth increases the potential reusability of knowledge and opens doors for improved productivity.

It is a common case that organizational knowledge was built in a small-scale early stage. This type of knowledge encounters challenges of scale-up, which is even more difficult to build up knowledge.

The growing organizations encounter new qualitative different challenges with new inter-organizational issues.

In this paper, the author proposed a two-dimensional model for knowledge support to deal with organizational dynamism. It highlights the importance of problem-domain-shift awareness. The time-dimensional and space-dimensional issues are highlighted using three-staged models. This facilitates the staged-awareness to open doors for organizational knowledge sharing in a new stage. Preparedness is important in knowledge system design as well as corporate management. Stage-transition support is an unexplored field in organizational knowledge support systems. It will increase importance in today's dynamic business environment. The proposed frameworks provide a milestone to move forward in facilitating organizational stage transitions.

References

[1] A. Ackerman and C. Halverson. Considering an organization's memory. In *ACM CSCW'96*, pages 39–48. ACM Press, November 1996.

[2] T. Yamakami. Information flow analysis: An approach to evaluate groupware adoption patterns. *Trans. IPSJ*, 36(10):2511–2519, October 1995.

[3] I. Nonaka and H. Takeuchi. *The Knowledge Creating Company*. Oxford Universtiy Press, 1995.

[4] Susumu Kunifuji. "Knowledge Resonance" in Knowledge Management (in Japanese). *Trans. IPSJ*, 47(9):1021–1027, September 2006.

[5] J. Kontio, J-P. Jokinen, M. Mäkelä, and V. Leino. Current practices and research opportunities in software business models. In *ICSE '05*, pages 1–4. ACM Press, May 2005.

[6] M. Cusumano and R. Selby. *Microsoft Secrets*. The Free Press, 1995.

[7] M. Cusumano. *The Business of Software: What Every Manager, Programmer, and Entrepreneur Must Know to Thrive and Survive in Good Times and Bad*. The Free Press, March 2004.