Software Development Industrialization by Process-Centered Style

● Jun Watanabe ● Tomiko Maruyama

(Manuscript received November 11, 2009)

Fujitsu Applications, Ltd. (FAP) was established in April 2004 as a development division based on Fujitsu's System Development Architecture & Support (SDAS) facilities with the mission of implementing SDAS and establishing/collecting development know-how. It is working on new technologies based on experimental and practical studies, including issues such as the separation of design and manufacture, the industrialization of software development, and the utilization of offshore resources. Moreover, by taking on the roles of a manufacturing factory and an experimental factory, FAP is promoting innovative production in the Fujitsu Group, including the development of technology from an on-site viewpoint, technology verification, and the establishment of methodologies for managing development. The software development industrialization effort can be regarded as a transition from an individually led development style to a process-centered systematic development style. This paper introduces our code of good practice for our autonomous improvement activities. This consists of four conventions and six mechanisms, which are described here with reference to actual results.

1. Introduction

Fujitsu Applications, Ltd. (FAP) is a member of the Fujitsu Group that specializes in the development of applications. We are chiefly involved in the development of enterprise applications, which are systems used for the day-to-day running of ordinary businesses and public organizations. We specialize in the use of Java as a development language, and we mainly develop Web-based applications. Application development currently faces three challenges: rapidly developing large-scale, high-quality systems at low cost, reducing overall life-cycle costs of systems, including their maintenance and development, and establishing a methodology for large-scale, decentralized development to facilitate recruitment and reduce costs by taking into consideration both domestic and overseas activities. FAP acts as a factory for enterprise applications, which it develops and maintains with the aim of satisfying the quality, cost, and delivery (QCD) standards presented by customers. At the same time, by concentrating on in-house production by supporting FAP's role as an experimental factory in driving innovative production in the Fujitsu Group, we are taking on challenges such as developing techniques based on an on-site viewpoint, verifying those techniques, and establishing a methodology for development management.

In this paper, we explain our efforts to industrialize the software development process—which is the central theme of innovative production—by taking steps toward improving processes from an individually led development style to a systematic process-centered development style. We consider the following four viewpoints.

1) Process-centered development method
2) Separation of design and manufacture, and document-centered development
3) Own-process completion
4) Streamlining

Next, we describe the four conventions and six mechanisms that constitute the code of good practice for on-site autonomous improvement activities. These are introduced together with lessons learned and problems experienced on site, including examples and case data.

2. Software development industrialization

FAP came to strongly acknowledge the importance of software development industrialization after finding that it is impossible to adapt to current demands using the development style of the general-purpose equipment era, where implementation concepts and technologies were jointly developed by development teams on the basis of each hardware vendor's architecture. As a result, the implementation technology was accumulated within each development team as tacit knowledge.

On the other hand, current implementations such as Java are based on frameworks. This means that correct knowledge about the implementation technology is required not just by the development team but also by the system's designers. To develop a large-scale system in a short period of time, it is essential to obtain contributions from a large number of people. However, it is impossible to gather large numbers of highly skilled and stable personnel in a short period of time. In the FAP practice example, there was found to be a productivity difference of 5.5 times[1] even when project members received the same initial training.

As a means of reforming things like the development style, implementation technology, and tacit knowledge architecture that depend on conventional high skill levels of specific individuals, we decided to turn our attention to industrialization. The aim of industrialization is to accurately adapt to diverse architectures (frameworks) and establish an orderly development style that does not depend on people. This is considered to be a means of aiming to reconfigure software production as a whole.

3. Dealing with industrialization

Efforts toward software development industrialization at FAP[2] have been greatly influenced by the Toyota Production System (TPS).[3] In 2004, we had the opportunity to participate in a TPS on-site guidance meeting, where the participants became aware that there are many concepts that can be applied to the software development site, including the clarity and ease of comprehending the target settings. As a result, we decided to concentrate on the two efforts of "human-centered management" and "thorough elimination of waste", and we set out to establish production methods and improve them constantly. FAP's industrialization efforts are a step towards the improvement of processes from an individually led development style to a systematic development style. They are overviewed in **Figure 1**.

3.1 Process-centered development method

A characteristic of the FAP production system is its use of a process-centered development method. As shown in **Figure 2**, FAP previously used a method that involved assigning jobs to people. The way of work depended on individual skill, so there was a big individual difference in productivity and the quality. On the other hand, process-centered work allocates jobs to standardized processes, so it is better able to bring about benefits such as eliminating individual differences, improving processes, and improving the standardization of work. In this way, it makes it easier to pursue overall optimization. However, this method requires skilled individuals to design processes,

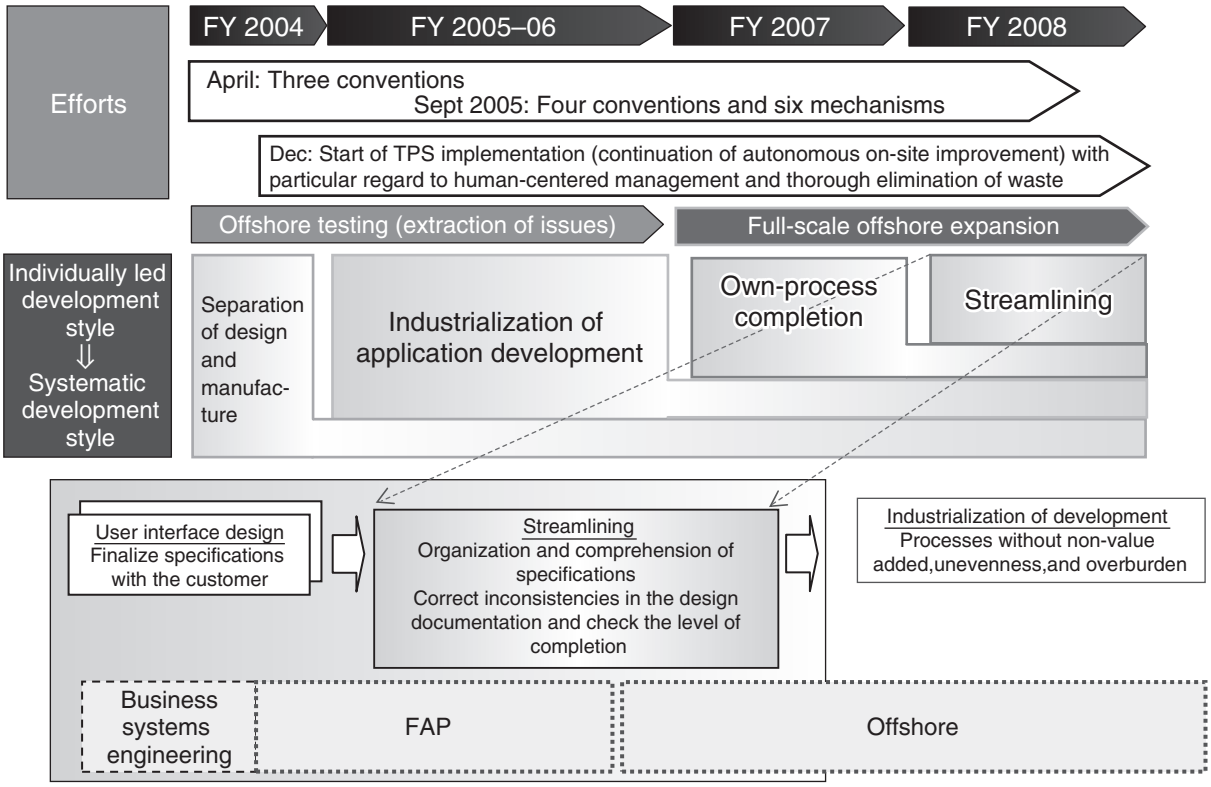
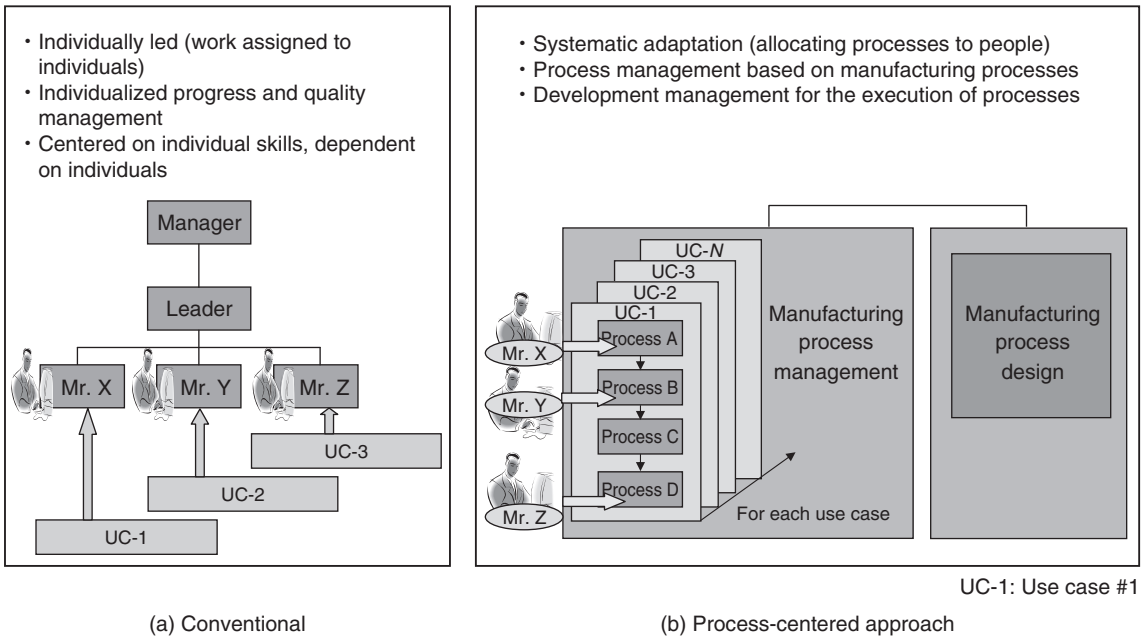


Figure 1
Progress in software development industrialization at FAP.



(a) Conventional

(b) Process-centered approach

Figure 2
Comparison of product development style at FAP.

and there is an urgent need to cultivate human resources capable of performing this process design. Another issue with the process-centered approach is that it tends to increase the cost of project preparation such as determining the details of processes, standardizing tasks, and training personnel. It is also important to apply the same process repeatedly.

3.2 Separation of design and manufacture and document-centered development

To operate a process, it is necessary to clearly separate design and manufacture. In the past, many things had to be defined and done, including process definitions, and the work breakdown structure called WBS. Even today these items are still not reliably separated.

With this factor taken into consideration, a style was devised whereby development proceeds while the designer and manufacturer conduct questioning and confirmation as required instead of inheriting the specifications from the same document or from clearly stated documents. Moreover, even in the conventional method of cultivating human resources by running apprenticeships, this sort of word-of-mouth transmission of specifications is achieved in the form of questions and eliminates opportunities for teaching abilities that promote understanding and writing skills that clearly express intentions in documents.

At FAP, we are constructing a work process that first establishes a clear separation of design and manufacture and calls for document-based handover as a basic requirement. Many people have pointed out this system's increased overheads and increased quantity of documentation and the imperfections in the transmission of specifications. However, we think that the separation of design and manufacture should be promoted by steadily overcoming those problems (that is, by minimizing overheads, reducing the amount of documentation, etc.).

3.3 Own-process completion

TPS includes the concept of own-process completion, whereby the quality of allocated processes is ensured to avoid any trouble in subsequent processes. To apply this concept to software development, we presume that each worker's quality responsibilities are made clear in the process subdivision and the separation of design and manufacture. If the quality responsibilities of each worker can be clearly stated, then it will become possible for each worker to work toward perfect quality in his or her own processes. In the case of process-centered development where a single program is handled by multiple individuals from design through to development, a thorough application of this concept is an absolute requirement for quality assurance.

3.4 Streamlining

In TPS, the word "flow" has a special meaning.[4] By adhering to this concept in work processes that are also prescribed in software development, we can consider a streamlining process that involves minimizing stagnation and reworking in order to make assets flow in a predetermined process sequence and in a standard period of time. By extracting and correcting the factors that disturb the regular flow of the manufacturing process by causing stagnation and reworking, we can ensure that the smooth running of subsequent processes is guaranteed. Currently, a streamlining process is where the finalized specifications are determined and fully understood, and any inconsistency or ambiguity in the specification documents that is discovered during these stages is confirmed and corrected. To achieve efficient streamlining, consistency check tools have been maintained. In addition, the pattern of matrix type's design document have been maintained.

4. Four conventions

At FAP, we have defined and implemented

four conventions and six mechanisms. These constitute a code of good practice for on-site autonomous improvement activities aimed at development process industrialization. The four conventions are used to raise the awareness of workers.

1) Measure labor in hours, not months

In the Japanese software industry, the amount of work and the contract value have traditionally been calculated using the formula *man-months* × *unit-cost* (i.e., the amount paid to one developer per unit time). This is akin to the concept of hiring people to be treated as objects and runs contrary to the human-centered management approach of TPS. The management unit of each process is time, and the smallest unit of time that is currently managed is 0.1 hours. The primary unit of cost is a few hundred thousand yen per man-month, but this corresponds to several thousand yen per hour, and changes in the on-site cost-consciousness lead to unnecessary cost-cutting.

2) Apply (record) work tickets correctly every day

Accurate cost appraisals are straightforward in hardware development, but they can be difficult in software development. At FAP, we ascertain value by using work tickets, which have been used for many years at Fujitsu.[5] These tickets let us not only ascertain the value of projects but also analyze work and conduct on-site improvements. The important point is to record the work tickets every day. If work tickets are not recorded every day but are lumped together into longer time periods, then the costs obtained on the basis of these tickets will be less accurate. The operation of work tickets is also implemented in offshore development, where management is implemented at the same level as in Japan.

3) Base discussions on data

At the development site, subjective communication such as "it will be finished soon" or "the quality is ok" are widely used and are often an excuse for fudging. This causes the individuals concerned to hold different opinions about the current situation. To prevent this, it is essential that everyone uses the same objective data and the same scale measures when taking part in discussions. Ensuring that discussions are based on numerical data is a basic principle of engineering.

4) Store information in specified places and in specified forms

We introduced the TPS idea of "sifting and sorting" into software development. The resulting merits are immeasurable. For example, if design documents are always stored in a folder with specified naming rules for any project, then it is possible for anyone analyzing the document to access the original data effortlessly whenever necessary.

5. Six mechanisms

The software production system at FAP is specifically configured from six mechanisms, as shown in **Figure 3**. Of these, mechanisms 1–3 are for operating the actual development and maintenance, 4 and 5 are functions supporting these mechanisms, and 6 is a production management system. Besides these mechanisms, there are overall management systems such as a human resource management system (skill management) and a cost management system (work tickets), which function cooperatively.

1) Function scale

In development using Java, there are many kinds of assets, and a framework is a prerequisite for software development. As a result, estimates based on the conventional scale estimate of the number of coding steps are judged to be impractical and have been completely replaced by estimates based on function scale measurements using the function scale method. More details are given in the paper "New Technique for Estimating Development Scale—Implementing the Function Scale Method" in this special issue.

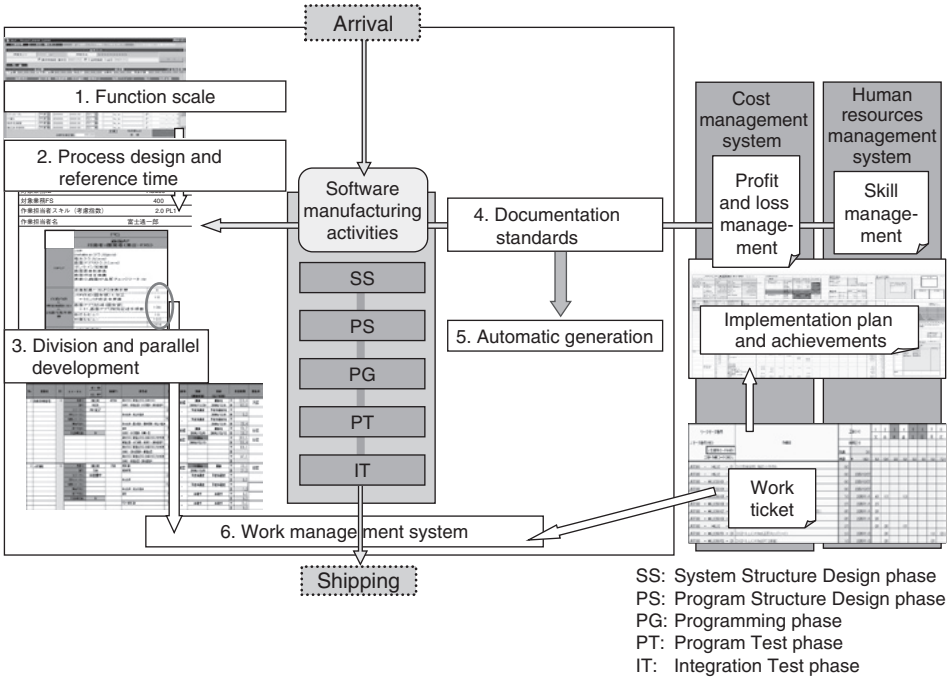


Figure 3
FAP software production system (six mechanisms).

2) Process design and reference time

At FAP, the concept of a process starts with dividing the work up as finely as possible, determining the inputs and outputs of each individual task, and setting the standardized work times. This process does not deal with individual people, but takes an overall view of the work including the corresponding administrators, reviewers, and configuration managers to form a large work map showing who does what where and showing how these individual tasks flow together.

We first started dividing work according to this idea in December 2004, and since that time, we have continued through a process of trial and error. An example of an actual work map is shown in **Figure 4**. This project was divided into 143 processes. The reference time, which is the time scheduled for a process by the person responsible for it, is calculated by multiplying the standard time by the skill coefficient and proficiency coefficient of the person doing the task.

3) Division and parallel development

In the detailed division of work and its expansion into a workflow, parts that can be developed in parallel are extracted. These days, shorter delivery times have a very high added value, so it is essential to improve this mechanism.

4) Document standards (design normalization)

A factor that is essential for the advancement of industrialization is design-level standardization. Since the separation of design and manufacture is taken as a precondition, industrialization will not function if there is instability in the originating designs and design documentation. At FAP, we are investigating the sufficiency of user interface design documentation in cooperation with Fujitsu. This involves checking the design documentation from a conventional design-oriented viewpoint, and it aims to achieve substantial improvements in both the breadth and depth of the design by judging

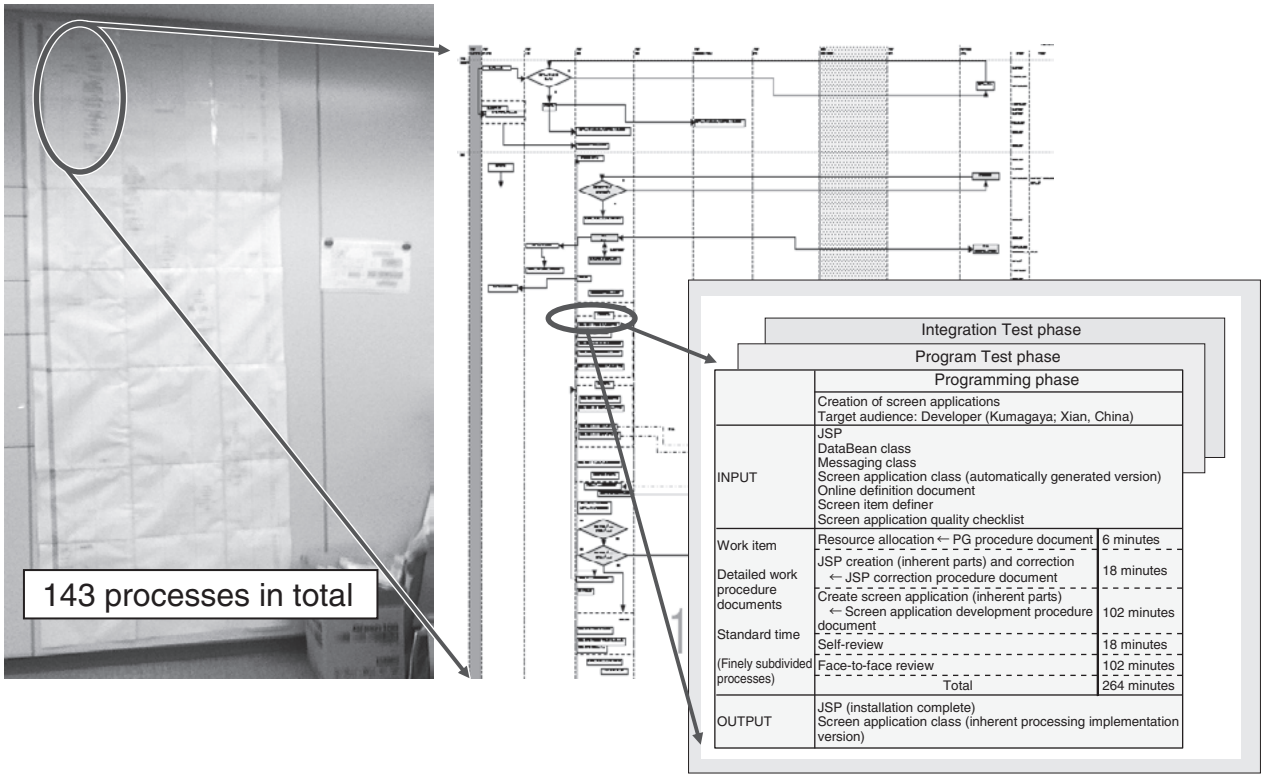


Figure 4
Process management map.

the design documentation from a manufacturing-oriented viewpoint.

5) Automatic generation

In manufacturing processes, parts that depend on automatic program generation are increasing. FAP has three policies related to automatic generation.

- Concentrate the efforts of workers on tasks that only humans are capable of doing rather than targeting them for automation
- Automate program generation with the aim of preventing the occurrence of simple mistakes in simple tasks
- Achieve automatic generation based on sifting and sorting (uniform creation), which is a weak point of human developers

At present, the rate of automatic generation is approximately 50–70%, including parts such as framework concealment. However, when software is generated automatically, it is important to ensure that the concerned individuals are aware of how far automatic generation will be taken and to clarify the role of human involvement. If this cannot be done thoroughly on site, then confusion is liable to ensue.

6) Work management system

This is a system for collecting the actual time from work tickets for processes and producing visualizations based on this information. It is not a simple software production management system, but also operates as a framework for rapidly inferring the issues (problems) that need to be addressed. The concept is illustrated in **Figure 5**. Achievements are inserted from work tickets for the reference time stated in the process, and any disparities are flagged for attention. Since processes are finely subdivided, progress and quality issues can be reliably spotted from reported discrepancies between the schedule and achievements for each process

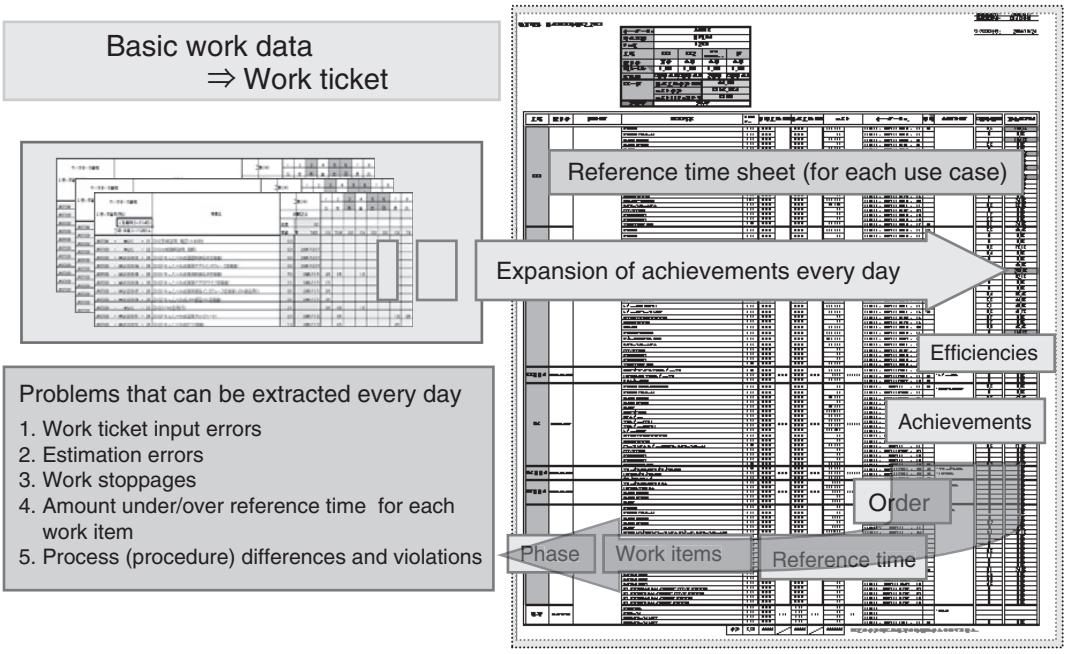


Figure 5
Concept of work management system.

instead of being inferred from people's subjective progress reports.

6. Development management

This section discusses the characteristics of ideas related to FAP management specialized for development. In software development, there are many issues requiring management, including standardization, reporting, quality, and configuration. It is essential that these are always implemented on the basis of a consistent technological approach. Development management is the continuation of governance for all individuals associated with development, and its main elements are architecture, standardization, and management techniques. For these elements, if governance is maintained continuously as an organization, it is possible to gain the benefits of technical accumulation and expertise and to accumulate development data. Having developed different business systems in a certain field of industry with the same architecture, the same standards, and the same management techniques with a 40% worker continuity rate, we increased productivity by a factor of 1.7 while achieving the same level of quality after shipping and a productivity dispersion of 1 part in 2.6.

Development environments are changing. Whereas development used to take place at a single location, we are now seeing distributed development and even offshore development projects. As this trend progresses, governance is expected to become increasingly important in development projects.

7. Conclusion

At FAP, we have tackled software development industrialization in various ways. This work is still incomplete and there are many areas requiring further work. Meeting QCD requirements is the most important theme, and key issues include improvements in employee satisfaction and practical implementation of an orderly development style.

A new topic that we are currently working

on is the analysis of the data accumulated so far. By collecting detailed data, one can amass over 100 000 records of achievement data upon the completion of a single project. We are working on the construction of a project risk estimation model based on this data.

Finally, progress in software development industrialization is the key to creating a desire for improvement and the ability to resolve problems in each individual worker on the scene. Problem-resolving ability requires a great deal of creativity, and cultivating this ability through the course of actual development/maintenance and development technology management is the most important theme.

References

1) T. Maruyama, et al.: Business application development through the introduction of the Toyota Production System. (in Japanese), *FUJITSU*, Vol. 57, No. 1, pp. 96–101 (2006).
2) J. Watanabe, et al.: Software Gemba Power Handbook. (in Japanese), 1st edition, Ohmsha, 2009, pp. 387–400.
3) T. Ohno: Toyota Production System—Beyond Large-Scale Production. (in Japanese), Diamond, Inc., 1978.
4) H. Iwaki: Practical Toyota Production System: A cost revolution to make the best use of personnel and organizations. (in Japanese), 1st edition, Nihon Keizai Shimbun, 2005, pp. 184–187.
5) J. Watanabe: (Practical implementations edition) Ensuring work tickets are supplied every day—the front line of the software factory. (in Japanese), ITpro, 2008. *http://itpro.nikkeibp.co.jp/article/COLUMN/20080528/304481/*

Jun Watanabe
Fujitsu Applications, Ltd.
Mr. Watanabe received a B.A. degree in Management from Dokkyo University, Saitama, Japan in 1974. Later that year, he joined Fujitsu Ltd., Tokyo, Japan. In 2004, he moved to Fujitsu Applications, Ltd., where he now serves as the President and Representative Director.

Tomiko Maruyama
Fujitsu Applications, Ltd.
Ms. Maruyama received a B.E. degree in Electronics Engineering from Ritsumeikan University, Kyoto, Japan in 1980. Later that year she joined Fujitsu Ltd., Tokyo, Japan. In 2004, she moved to Fujitsu Applications, Ltd., where she has been engaged in the development of enterprise business applications.