Diffusing Software-based Technologies with a Software Factory Approach for Software Development

A Theoretical Framework

Lim, Ngang-Kwang, Ang, S. K. James, and Pavri, F.N.
fbap7630@nus.edu.sg, fbaangsk@nus.edu.sg, and fpavri@nus.edu.sg
Department of Decision Sciences, Faculty of Business Administration,
National University of Singapore, Republic of Singapore

Abstract

We present the theoretical framework of a research currently being conducted to formulate a Software Factory model for diffusing software-based technologies to attain high software throughput. First, we review three existing Software Factory approaches, and then extract relevant attributes from these approaches to formulate our theoretical framework. We focus on three major components involved in the software development process. These components comprise both management and technical aspects. They are 'technology' and business management, software engineering environment, and software engineering process. Using this framework, we are now in the process of developing a practical, disciplined Software Factory model.

Keywords: Software Factory, 'technology' and business management, software engineering environment, software engineering process

BRT Keywords: IB02, FC15, FA

Introduction – Background

Developing East Asian countries have placed heavy emphasis on acquiring technological competency, a key to generate economic growth (Asian Development Bank, 1995). With the rapid growth of the Information Technology and software industry, software technology would be a key technological competency for these developing countries (World Bank, 1989). Attaining substantial level of software throughput would translate to increase in exports, thus contributing to a higher economic value for the country.

Raising software throughput can be achieved through diffusing software-based technologies in a massive scale with a robust, consistent and efficient means of producing quality and reliable software. A good approach for such software development process is the Software Factory concept, developing software in a production-like manner and within a factory-like environment.

The concept of a Software Factory denotes a software production environment that supports a more industralised production of software with engineering-like and manufacturing-like practices. While the concept of a Software Factory Process was heavily researched over the past 30 years, the adoption and application of such process has not yet reached a massive scale.

Our initial efforts to formulate the theoretical framework for a Software Factory model were drawn from work done by Evans (1989), Cusumano (1991) and Matsumoto (1989), and Weber (1994) [see Table 1]. Collectively, their work represents a global view of the Software Factory process. Our efforts were carried out independently of Aaen, Bottcher and Mathiassen (1997).

Evans' Approach	**Cusumano and Matsumoto's Approach**	**Eureka Software Factory**
- quality software - predicatable, controlled and productive fashion	- mass production - large-scale centralised operations - standardised control	- industrialised approach

Table 1 : Software Factory Approaches

The software factory concept dates back to the 1950s and late 1960s when it was first proposed in the United States and Europe. The concept was originally proposed in United States to improve software development productivity through standard tools, methods and component reuse (McIlroy, 1959 ; Bemer 1969). In the mid 1970s to the mid 1980s, Japanese major computer manufacturers furthered the concept, making significant improvement in software development (Cusumano, 1989 ; Cusumano, 1991 ; Matsumoto, 1989).

In 1987, the Eureka Software Factory project with a 7-year timeframe was launched with the objective of improving large-scale software production by introducing an industrialized approach where advanced technology and novel ideas in process engineering were applied (Weber, 1997). By early 1990s, substantial research on the Software Factory Process has been reported (e.g., Cusumano, 1991 ; Weber, 1997).*'What is now known is that Software Factories can be built, the approach is viable, and there are industry and research groups who know how to do it. Only through careful investment, will we see Software Factories in general use.'* (Weber, ed., 1997, p. 213).

Formulating the Software Factory model to achieve diffusion of software-based technologies

Taiwan, an East Asian developing country, has been very successful in diffusing technologies and spinning off technology start up companies in the computer and semiconductor industry (Gee, 1990). In Taiwan, specific government-linked research institutions were formed to conduct research and development in electronics industrial technology, and then disseminate the acquired technology to both the public and private sectors (Philippe, 1996) (Figure 1). One such institution, ITRI (Industrial Technology Research Institute), has a clearly defined mission to provide advanced technology support to Taiwanese industry and assisting small firms to overcome barriers to entry (Rush, 1996).

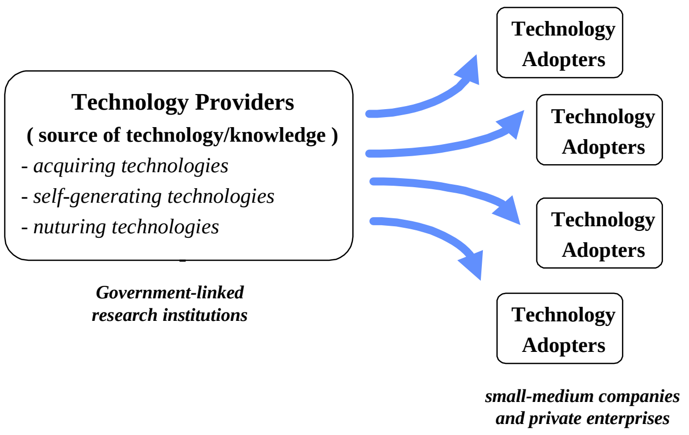
Technology Providers
(source of technology/knowledge)
- acquiring technologies
- self-generating technologies
- nuturing technologies
Government-linked research institutions
Technology Adopters
Technology Adopters
Technology Adopters
Technology Adopters
small-medium companies and private enterprises

Figure 1 : Diffusion process in hardware and semiconductor high-tech industry

Such technology diffusion mechanism could be extended to the software industry, and research institutions or organizations that develop and diffuse software-based technologies to technology adopters (particularly small and medium companies) could also adopt a Software Factory model (Figure 2). Software development and diffusion would then be carried out in a highly productivity, predictable and consistent manner.

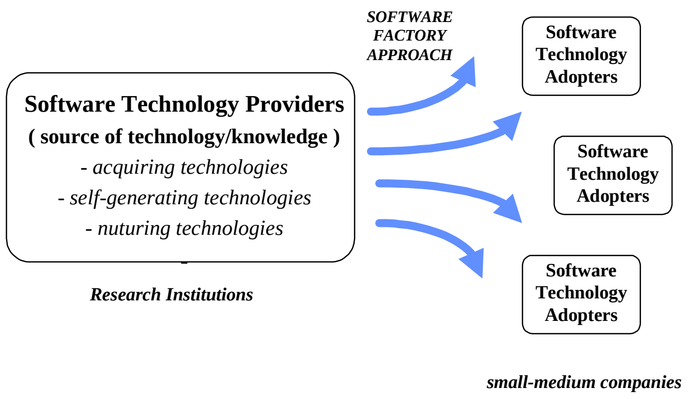
SOFTWARE FACTORY APPROACH
Software Technology Providers
(source of technology/knowledge)
- acquiring technologies
- self-generating technologies
- nuturing technologies
Research Institutions
Software Technology Adopters
Software Technology Adopters
Software Technology Adopters
small-medium companies

Figure 2 : Proposed diffusion process for software-based technologies

In a manufacturing environment, all the facets of manufacturing, the interfaces and overlaps within manufacturing, are described by the traditional 5Ms and 2Ss system (Whitmyre, 1987).

- . Manpower : people in production
- . Methods : designing, improving, and controlling production systems
- . Machines : production processes and equipment
- . Material : asset management
- . Money : financial management
- . Space : the production facility
- . System : factory integration

In our theoretical framework, the 5Ms are replaced by the three major components of our Software Factory model :
- 'Technology' and Business Management for 'Manpower', 'Money' and 'Material',
- Software Engineering Environment for 'Machines', and
- Software Engineering Process for 'Methods'.

'Technology' and Business Management manages the process of how the software-based technologies are generated and diffused to the technology adopters, and manages the business drivers and business conditions, organizational infrastructure and organizational design.

Software Engineering Environment is the engineering environment, the system and the tools that are used for generating the software programs.

Software Engineering Process is the engineering process of generating, controlling, and maintaining the generation of the software programs.

The Software Factory Process

Evans' vision of the Software Factory Process (Fourth Generation Software Engineering) is an environment in which quality software can be produced in a predictable, controlled, and productive fashion, approaching that of an assembly line - a 'Software Production Assembly Line'. To Cusumano, the Software Factory Process refers to factory-like concepts and technologies. It is a process which entails mass-producing products with large-scale centralized operations, standardized and skilled job tasks, standardized controls, specialized but low-skill workers, divisions of labour, mechanisation and automation, and interchangeable parts. Weber's Eureka project sought to improve large-scale software production process by introducing a disciplined, industrialized approach.

We extracted relevant attributes from these efforts, and used them to formulate our proposed Software Factory model. Each of the three components of the Software Factory model is a discipline on its own and has an in-depth body of research. In fact, the attributes of each component have their own established industrial standards (Moore, 1998). The grouping of the attributes into three major components (or categories) permits the model to be used with ease. Attributes that are similar in nature are grouped together (see Table 2). Each component is explored in detail :

1. ‘Technology’ and Business Management
   - division of labour and formation of functional groups
   - project management
   - specifying performance

2. Software Engineering Environment
   - documentation
   - data control and management
   - tools standardization
   - automated environment
   - product assurance

3. Software Engineering Process
   - process standardization
   - systematic reusability of codes (re-cycling software designs or codes)
   - productivity measure
   - quality measure and defect control

Manpower, Material, Money **‘Technology’ and Business Management**	***Machines*** **Software Engineering Environment**	***Methods*** **Software Engineering Process**
- division of labour and formation of functional groups	- documentation	- process standardisation
- project management	- data control and management	- systematic reusability of codes
- specifying performance	- tools standardisation	- productivity measure
	- automated environment	- quality measure and defect control
	- product assurance	

Table 2 : Essential components of a Software Factory Process

On ‘TECHNOLOGY’ and BUSINESS MANAGEMENT

An appropriate organizational infrastructure is critical to the success of a Software Factory (Cusumano, 1989). For example, a rigid separation of design from product construction is likely to be counter-productive if organizational issues are not addressed. A Software Factory, in fact, consists of : 1). A Factory Organization, and 2). A Factory Support Environment. In adopting the Software Factory Process, organizational changes are expected, and top management must be committed to support these changes. A long-term commitment is needed as most of these changes take time to reach steady state.

Evans' Work

Evans focussed on a two level organization. Level 1 (the top management level) directs, monitors, controls, and supports the software project. It also provides a focal point for all accountability and management responsibility. Level 2, the secondary level, provides technical development and support.

* Level 1 - management groups. These groups focus on the business aspect. They ensure that the project is managed in accordance with the contract, and that the budgets and schedules are properly identified, allocated, and monitored. These groups also maintain the contract database, and control and manage 'change review boards' and problem reporting functions.

The groups start with the software project manager who has overall responsibility for the project, including all project activities, the definition of an engineering environment adapted to the project, and technical management. Reporting to the project manager are the product assurance group and the validation and verification group.

* Level 2 - technical development and support groups. These groups focus on the technical performance, and support the technical and control requirements of the project. The two major groups are the software system engineering group and the software development group.

The software system engineering group establishes the project's operational concept and provides technical support to Level 1 on all system and user interface issues. This group develops, translates, and validates user scenarios, defines the automated, procedural, and control requirements, and develops, document, and enforces the standards established for the software engineering environment.

Cusumano and Matsumoto's Work

"Japanese software factories were strategic in that they resulted from long-term rather than just project-centered objectives, with managers systematically identifying productivity, quality, and performance goals, and introducing comprehensive measures to meet those goals" (Cusumano, 1991, p. 425). The managers had to link their products and marketing strategies with organizational structures, production technology, control systems, training, and other elements. Implementation requires leadership from high- and middle-level managers as well as senior engineers, and the allocation of enough resources to insure that organizational experimentation as well as process research and development will continue.

Cusumano's studies (1991) on major Japanese manufacturers emphasized these areas :

* Commitment to Process Improvement :

The managers demonstrated a long-term commitment to process improvement, and believed that they could structure and improve software operations and achieve higher, or more predictable, levels of productivity, quality, and scheduling control. As there is a need to allocate time and engineering resources to study many projects, build tools, train personnel, or develop reusable designs and code, it is necessary to have commitment from top managers.

* Skills Standardization and Leverage :

The focus is on the improvement of capabilities for process and quality standardization. Standardization of skills primarily through extensive training of all new recruits in a set of standardized methods and tools helps raise the average level of productivity. This should be done without creating a completely rigid process for software engineering, or constrain an organization and individual projects from meeting the needs of different customers or design efforts.

* Tailored and Centralized Process R&D :

Organizations adopting the factory approach have centralized tool and methodology R&D at one level above the individual projects, so that all projects have equal access to good tools and techniques. Process R&D are also centralized at the product-division or facility level rather than at the corporate level, and this helped to accommodate the needs of different product types.

* Product-Process Focus and Segmentation :

The factory approach is to segment products and processes by channeling non-routine work into less structured modes of production. Particular types of software products are focussed at the facility or division level, and processes like methods, tools, standards, training, are gradually tailored to these product families and particular customer segments. Process segmentation allows managers to channel similar work to specialized groups while sending new or non-specialized jobs (for which the organization had no special accumulation of skills or investment in tools) outside the factory.

* Incremental Product and Variety Improvement :

The Japanese software producers first concentrated on producing software more efficiently and guaranteeing product reliability to customers. After demonstrating high levels of productivity and reliability, they then gradually turn to upgrading product designs and performance. They spent increasing amount of time in design rather than in routine tasks such as coding, and expanded the variety of products they produced in a single facility.

Eureka Software Factory

The Eureka project viewed the Software Factory as a business or industrial company, or as a profit center. As such, the company infrastructure is under the control of a management group operating in a changing environment. The infrastructure includes organizational definitions and support-environments, namely :
- knowledge and descriptions of all critical activities in the organization
- support for managerial, administrative, business and technical activities
- integration mechanisms providing coordination across all of these activities or support facilities

Key roles being defined by the Eureka Software Factory include : System Manager, Method Manager, Project Manager, Designer, Programmer, Quality Controller, and Product Manager.

On SOFTWARE ENGINEERING ENVIRONMENT

Evans defines the software environment (including automated facilities) as one involving the "*implementation of development policies and practices, and is the means by which development requirements are translated into project activities. When implemented, these practices are used to develop and maintain software data products.*" (Evans, 1989, p. 32). Besides adopting technology to integrate all software development activities, crucially needed are : standardization of product and process, of management planning and control, and the integration of software and system development.

Evans' Work

Evans described the Software Factory Environment as a Software Engineering Process at the core with three constituents surrounding it. They are : 1) data management, standardization, and control, 2) engineering technology and control, and 3) product assurance and management control. He emphasized on Data Control and Management, adding that the *"integration of the software engineering environment should focus on the data bridges between separate engineering activities"* (Evans, 1989, p. 32).

With regard to technical requirements, each technical requirement demands its own solutions to ensure that it is adequately supported. The technical solutions fall into the following project development categories : 1) Management Methods, 2) System Requirements, 3) System Design, 4) Prototyping, 5) Software Design, 6) Software Coding, 7) Software Development Testing, 8) Software Testing, 9) System Testing, 10) Data Production, 11) Baseline Configuration Management and Control, 12) Engineering Data Management, 13) Quality Control, 14) Project Monitoring and Control.

* <u>Automated Environment</u> :

The automated part of the software engineering environment is a complete, layered structure of tools, data structures, and access techniques and controls. This environment is constantly evolving. The tools are used in defining requirements, creating, controlling, and verifying the coding, and finally, altering code and design requirements.

The automated environment consists of a series of layers or components : automated support, data management, configuration management, support tools, communications, security, local access, and workstation support.

* <u>Product Assurance</u> :

Product assurance is an integral part of the engineering environment. It provides technical and quality gates for all information used during software development or delivered to a higher-level life cycle or to the customer. Product assurance methods provide ongoing assessment of the effectiveness of the environment, the development risk, productivity, and project performance.

Product assurance includes :
a) certification to assure the integrity of the product against acceptability criteria, and
b) integration and test procedures.

Cusumano and Matsumoto’s Works

All the Japanese manufacturers, studied by Cusumano (1989) and Matsumoto (1989), relied on mechanisation and automation - CASE (Computer Aided Software Environment) tools for product development, project management, data collection and analysis, reuse support, and quality control. These tools capture expertise, reinforce good practices, allow personnel to spend less time on routine tasks, and make it possible for relatively unskilled personnel to build complex, unique or customized systems.

To make tool usage more efficient and coordinated with a standard development process, Japanese manufacturers also pay considerable attention to integrating tools with each other as well as with standardized methods. They also pursue better tools and methods continually through some form of R&D organized above the project level.

The CASE (Computer Aided Software Environment) environments in general have the following five hierarchical levels (Matsumoto, 1989) :
1. national level (e.g., SIGMA in Japan)
2. company level (e.g., CASE center of the company)
3. factory level (e.g., CASE subcenter)
4. department or section level (e.g., cluster)
5. individual level (e.g., engineering workstation)

Items 1,2 and 5 are self-explanatory.

The factory level computer manages software assets such as software products which have been accepted by customers and are in operation at the customer sites. The factory level computer also serves to provide management with information such as factory-measured productivity and quality.

At the department or section level, an example of a CASE environment is to have clusters or groupings of workstations. The functions performed by a cluster are :
- to support management of software configuration items which are commonly used in the department or the section, and
- to support management of progress scheduling, cost, resource expenditure, project productivity, personal experiences and software quality which the manager of the department is responsible for.

Eureka Software Factory

As pointed out earlier, the Eureka Software Factory consists of : 1) a Factory Organization, and 2) a Factory Support Environment, which is the computerized part of the Software Factory. The architecture of the Factory Support Environment is a broad architectural framework which embraces the range of methods and tools specific to life-cycle support and be able to evolve in steps with commercial and industrial trends.

The technology requirements comprise two specific fields of technology - Componentry and Process Technology. Componentry seeks to construct the Factory from standard re-usable parts, reducing construction costs, and developing or modifying integrable components. Process Technology centers on the Process Model, which defines the roles played by the users of the factory, the information and services they need to do the tasks, and the lines of communication with other processes in the Factory.

The Eureka Software Factory's target is to configure components into a Software Factory Environment using a communication-centered architecture. The elements that interrelate in the Factory determine four levels of communication :
- inter-working between people engaged in group activity, or between groups of people
- interaction between the specific roles undertaken by people and the tools and services required in performing each role, for all roles which are required in the Software Factory
- inter-connection between the IT platform in local or distributed configurations
- inter-operation of the tools and services in the Factory Support Environment.

On SOFTWARE ENGINEERING PROCESS

The third critical aspect of the Software Factory Process is the Engineering Process, which has to be a consistent production process, producing software in a consistent manner with continuous improvement, effective statistical process control, and a high level of predictable result.

Evans' Work

Evans' definition of the engineering process is one *"used to identify and analyse requirements, design code, test software, and produce and release data"* (Evans, 1989, p. 64).

The engineering process has six essential elements :
1. Methods - the processes used to accomplish development, management, or support tasks
2. Data Relationships - the engineering data relationships that link the development, management, and support methods.
3. Data Control : the Configuration Management methods, procedures, and controls that ensure the integrity of the information used, developed, or supported by the software environment.
4. Software Data Base : the content, structure, and organization of the database that manages and provides access to information controlled or required by the software environment.
5. Verification : the policies that ensure the integrity of information of software released from the software environment.
6. Testing and Integration : the strategies used to ensure the integrity of released products.'

A method provides a consistent means to accomplish a task in the software engineering environment. It describes an engineering activity and standardizes the engineering, management, or data products that are developed. The methods used by a project establish policies and standards that will support the engineering process. The methods link the engineering, management, and support components of the software engineering environment. They also provide a common basis for estimating and applying resources and for monitoring and controlling their effectiveness.

Cusumano and Matsumoto's Work

The studies by Cusumano (1989) and Matsumoto (1989) revealed that Japanese firms are effective in adopting factory-like concepts of standardization of skills, productivity measurements, quality measure and defect control, and product assurance. Reusability of codes is also highly adopted both for productivity gain and quality improvement.

* Dynamic Standardization :

Standards are imposed for personnel performance, methods, tools, products, training, and other operations. The process of periodically reviewing and changing standards is also formalized. The policy of reviewing and revising standards for practice and performance ensures that an organization moves forward with the technology, and retains the ability to adapt to evolving customer needs.

* Process/Quality Analysis and Control :

Achieving greater predictability in cost and scheduling, as well as in quality, necessitates the introduction of performance standards and controls for every phase of engineering, testing, and project management. Companies could standardize personnel skills and product quality through a product focus and a standard process, as well as training in standard sets of tools, methods, and management procedures. There is also a need to invest extensively in data collection and analysis on the development process for each product family, so as to accumulate knowledge about products as well as discovering the appropriate tools, methods, and components.

* System Reusability :

The design and mass production of interchangeable components is another key element to the success in implementing the Software Factory Process. Planning and devising tools, libraries, reward and control systems, training techniques, are needed to maximize the writing of reusable software and the systematic reuse of components across different projects. Design for reuse in particular constitutes an investment in an ever-expanding inventory of reusable parts, though reuse is especially difficult across different kinds of software.

* Productivity Measure :

Organizations adopting the Software Factory Process is expected to attain specified quality and productivity levels. Productivity measurement is classified into four classes (Matsumoto, 1989) :

- price-based productivity : the total revenue of the software products which have been officially accepted in the fiscal year
- profit-based productivity : the total profit which has been made by the software products which have been officially accepted in the fiscal year

- cost-based productivity : the total costs which have been expended in producing the software products which have been officially accepted in the fiscal year
- capability-based productivity : the total number of equivalent assembler source lines of the software products which have been officially accepted in the fiscal year.

Eureka Software Factory

The Eureka Software Factory emphasized that the engineering processes should be modeled and enacted via the Factory Support Environment to guide or govern a co-operating team towards an agreed goal. It highlighted the need for a Process Model, which is defined as a system description of the software development organization, with process defined as a systematic series of human and/or automatic actions needed to accomplish some tasks. The tasks are supported by computer, for the definition, enactment and monitoring of processes.

Model of such software engineering processes is used by the run-time mechanisms of the Factory Support Environment to provide assistance to the individuals taking part in the processes. The types of assistance include these following forms :
- Process guidance, which provides individuals with information of what is expected of them at any point in time.
- Process assistance, which provides individuals with task-related work environments, including the management of information flow between individuals so that correct and appropriate information is available when needed, and the management of tools and their linkage with the data to which they will be applied.
- Process enforcement, which consists of constraining users to perform processes or part-processes in conformance with the process model.
- Process automation, which consists of automatically performing some part of a process without the need for human intervention.

The processes are modeled using the knowledge of the people in the organization and the procedural documents they used. An understanding of such models can improve the understanding of the process by those people involved in its performance, giving the potential to improve the overall group performance by refining the process itself.

Framework of the Software Factory Model

The above provided an insight into the work of Evans, Cusumano and Weber on Software Factory processes. We drew on related and relevant attributes from their research, and use them to build up the theoretical framework for our proposed Software Factory model. The framework involves three essential components identified for the Software Factory Process. Organizations planning to adopt the Software Factory model can use this framework to develop the model.

	Software Factory Process Attributes		
	Evan's	**Cusumano & Matsumoto's**	**Eureka Software Factory**
Organisational Infrastructure, Business Management and Control	* Two level organisation : management level and technical development & support groups	* Management commitment to process improvement * Skills standardisation and leverage *Tailored and centralised process R&D * Product-process focus and segmentation * Incremental product and variety improvement	* Infrastructure includes organisational definitions and support-environments * Key roles defined for each functional role in the factory
Software Engineering Environment	* Data control and management * Automated environment * Product Assurance : - Certification - Integration and test	* Computer aided software environment (CASE) * CASE : factory level, department level, individual level	* Componentry : - software reuse - communication among groups
Software Engineering Process	* Methods * Data relationships * Data control * Software database * Verification (reviews) * Testing and integration	* Dynamic standardisation * Processs/quality analysis and control * System reusability * Productivity measure	* Process Model : Process assistance to individuals - process guidance - process assistance - process enforcement - process automation

Table 3 : Components of the Software Factory Framework

Building the Software Factory

Along the proposed framework for developing a Software Factory model, we also propose the next step : the process of building the Software Factory. In the Eureka Software Factory project, it was pointed out that the Software Factory has to be built according to specific requirements depending on the organization, methods of work, and the industrial domain of the enterprise. These are related factors that need to be taken into consideration when building the Software Factory.

The building process is outlined below. In doing so, we adopt some recommendations from the Eureka Software Factory project. The Software Factory has to evolve constantly, taking into account changes in the environment, the business needs, and the organizational changes.

The building process steps are :
1. Preliminary study of the adopting organization to understand specific requirements and business directions.
2. Delineation of management, business and organizational boundaries.
3. Review of appropriate Software Engineering Environment (Software Factory) to be adopted.
4. Definition of the Software Engineering Processes (Software Factory) to be used.
5. Roll-out to organization and staff.
6. Setting up the Software Engineering Environment and Software Engineering Process, including staff training in the use of tools and processes.

This building process gives a generic outline for adopting organizations to follow. For established software development organizations whereby an established software development process is already in place, the building process steps need to be adjusted accordingly for the particular organization.

Measuring the Software Factory Process

Once the Software Factory model is in place, measurement matrices would be needed to measure the benefits of adopting such model. This could be achieved by monitoring and measuring the productivity and quality levels, which are measurement matrices typically adopted by software organizations.

Both measurement matrices have their own industry standards (Moore, 1998), and extensive research has been done on what is the most appropriate measurement methods and measurement units to adopt. By adopting measurement units proposed by Caper Jones (1993), it allows benchmarking to be made between developing and well developed countries, as regular surveys on these benchmarks are done and the results published.

For productivity measurement, we propose the net productivity level, *with 'net' defined as a fine metric aimed at evaluation of specific projects, deliverables, activities, and tasks* (Jones, 1993). The net productivity level is calculated by taking the total delivered feature points, and divide it by the specific amount of efforts devoted to producing them. The units of measurement would be : **delivered feature point per staff work month**.

For quality measurement, **delivered defects per feature point** is proposed. The term ‘delivered defects’ is the quantity of latent defects still lurking in the software when it is first delivered to the next level of end-users. *The quantity of delivered defects is much larger than the quantity of defects found and reported back during the first year of customer usage. Normally from 30% to 70% of delivered defects will be found and reported in the first year of usage* (Jones, 1993).

Conclusion

With substantial research already done on the Software Factory approaches, and various large industrial manufacturers like those in Japan (Cusumano, 1991) adopting the process in one form or another, the challenge would be to extend the application of such concept.

With our theoretical framework, we want to formulate a Software Factory model for diffusing software-based technologies. What we have done so far is to capture and consolidate the related and relevant attributes of Evan’s, Cusumano’s and Weber’s work on Software Factory approaches, and set up a theoretical framework to aid our line of thought. We have also laid down the steps for building a Software Factory and proposed measuring matrices for benchmarking.

We have since embarked on research activities to formulate a practical Software Factory model based on this theoretical framework. As a start, we have worked with a major R&D organization involving in diffusing software-based technologies. We adopted dual research methodologies : grounded theory (Glaser and Strauss, 1967) to frame the organization’s software development process according to the Software Factory framework, and action research to induce changes in the organization’s software development process to attain that of the Software Factory model. We are making good progress, working towards building up a preliminary Software Factory model.

Organizations planning to adopt a Software Factory model for diffusing software-based technologies could review the above mentioned components and attributes of the proposed framework, align and fine tune the framework accordingly, build the Software Factory, and measure the attainments. Once this is demonstrated, subsequent software development projects on new cycle of technology in similar industry domain or new software products to be developed, will relentlessly flow through this set-up. The proposal could be nutured upwards from an enterprise level to an industrial level, and towards a National level, attaining a platform for both technology and economic growth.

References

Aaen, I., Bottcher, P. and Mathiassen, L., “The Software Factory : Contributions and Illusions”, in Information Systems Research Seminar in Scandinavia, IRIS 20, 1997.

Asian Development Bank Publications, “Technology Transfer and Development Implications for Developing Asia”, May 1995.

Bemer, R.W., “Position Papers for Panel Discussion : The Economics of Program Production”, in Information Processing 68, North-Holland, Amsterdam, 1969, pp. 1626-1627.

Cusumano, Michael A., “The Software Factory - A Historical Interpretation”, IEEE Software, March 1989.

Cusumano, Michael A., “Japanese’s Software Factories - A Challenge to U.S. Management”, Oxford University Press, 1991.

Evans, Michael W., “The Software factory : A Fourth Generation Software Engineering Environment”, John Wiley & Sons, 1989.

Glaser, B. and Strauss, A.L., “The Discovery of Grounded Theory : Strategies for Qualitative Research”, Aldine, 1967.

Johnson, James R., “The Software Factory - Managing Software Development and Maintenance”, QED Information Sciences, Inc., 1991.

Jones, Capers, “Software Productivity and Quality Today : The Worldwide Perspective”, Information Systems Management Group, 1993.

Matsumoto, Yoshihiro and Ohno, Yutaka, “Japanese Perspectives in Software Engineering”, Addison-Wesley Publishing Company, 1989.

McIlroy, M.D., “Mass-Produced Software Components,” in Software Engineering : Reports on a Conference Sponsored by the NATO Science Committee, P. Naur and B. Randell, eds., Scientific Affairs Div., NATO, Brussels, 1969, pp. 151-155.

Moore, James W., “Software Engineering Standards - A User’s Road Map”, IEEE Computer Society, 1998.

Philippe, Lesserre and Hellmut, Schiitte, “Strategies for Asia-Pacific”, Macmillan, 1995.

Rush H., Hobday M., Bessant J., Arnold E., Murray R., ‘Technology Institutes : Strategies for Best Practice’, International Thomson Business Press, 1996.

San, Gee, “The Status and an Evaluation of the Electronics Industry in Taiwan”, Technical Papers No. 29, OECD Development Centre, 1990, pp. 31.

Schware, Robert, “The World Software Industry and Software Engineering. Opportunities and Constraints for Newly Industrialized Economies”, The World Bank, Washington DC, 1989.

Weber, Herbert, (editor), “The Software Factory Challenge - Results of the Eureka Software Factory Project”, IOS Press, 1997.

Whitmyre, David W., “The Manufacturing Enterprise” in White, John A., “Production Handbook”, John Wiley & Sons, fourth editiion, 1987.