New Trends in Software Methodologies, Tools and Techniques
H. Fujita (Ed.)
IOS Press, 2010

doi:10.3233/978-1-60750-629-4-415

Development of a Commercial Product Including Software

Zenya KOONO[a,1] and Hui CHEN[b]
[a]*Creation Project, Japan*
[b]*Center for Information Science, Kokushikan University, Japan.*

Abstract. Current software engineering assumes that software is developed to meet the requirements of clients. Apart for this business model, there has been another and more universal business model. Namely, a person or a company develops their own commercial product/service of pure software or combined software and hardware, delivers it to a free and competing market and receives the revenue. Major systems vendors, big software vendors and embedded system vendors have been working with this business model. This paper explains the activities in the creation phase of the product or service, management requirements and development work in the new business model.

Keywords. design, development, software engineering, systems engineering, commercial product, intentional activity

Introduction

The most fundamental business model is to manufacture and sell a commercial product or to provide a service (hereafter also a product) and then to receive the revenue such provides. A commercial product must win against opponents in a free and competing market. *Product might* be able to govern the result. Major factors of *product might* are functionality (including the power rating), price, delivery, quality and so on. "Marketing" focuses on the target customers, then "product planning" establishes a model to fit the target, and then "design" and "production" are carried out so as to create the best product with strongest *product might*.

Each product must be updated from several monthly intervals to several yearly intervals. Considering the framework of each business, a strategic plan of each product is devised initially, taking into consideration each product cycle and technical trends. Based on this strategic plan, a budget for the sales returns, profits, and research and development costs is reviewed and adjusted to fit the relevant status fixed for each (4 months or yearly) budget period. A new product must be developed according to a plan, and the targeted sales and profit must be achieved as planned. New product development is a very tight and demanding work. As competitors make similar efforts, the only thing to do is to fight against and try and defeat them in the next phase product line. Thus the execution plan of each new product is revised as to win in the market. Let us suppose that the targeted values were achieved, the team members receive a

[1] Corresponding Author: Representative, Creation project, Honfujisawa 2-13-5, Fujisawa, Kanagawa, 251-0875, Japan; E-mail: koono@vesta.ocn.ne.jp; URL: http://www.creationproj.org/.

corresponding bonus, and the future prospect of each promotion might be perceived. In the area of emerging technology, when the improvement brought about by technological advance are included also, most updates improve the *product might*. Engineering work is tight but the rewards make it worthwhile, and those involved are proud to be part of it.

In the enterprise system period, the aim of the business model was to convert what a client required into software. The core part of future IT systems is strategic systems, in which top management provide the leadership for the development[1], and the main part will be developed in their own company. This situation is the same as in an embedded system; the company involved should develop the central part. The business model will change from the notion that "software is a conversion derived from clients' requirements" to "software is created by those who have the business responsibility," as is the case with engineering in general.

This paper discusses the business model for software system development, where those who have the business responsibility create software and the model will be explained through practical examples. In Section 1, "Lessons learnt from hit products", 8 success stories are included. This paper summarizes the main points of each success, the approaches to new technology, the hard work that went into creation, what and how to learn from customers, and various management strategies to create hits, and so on. These have not been examined in previous business models. More than the differences in technology, the different circumstances of each situation will appeal to readers. Section 2, "Requirements from top management" explains major requirements and strategies of top management, and evaluates quantitatively the current status of previous business models. Software people in previous business models have not realized what requirements are essential for success. Section 1 may be like a show window displaying many stars of success, while Section 2 is like a list of their limitations, each of which has some! Section 3 traces the path from the start of design to the actual development process where all the sections or people participate and work together for the success of a commercial product. A success is not an occasional happening but the fruit of a systematic process of combining various abilities and effort.

1. Lessons learnt from hits

1.1 Successes in conventional areas

In modern society, most of the wealth is created by industry, and their products and services (hereafter denoted simply by products) contribute to the world, the nation, the company, and then to the employees and their families. Engineering is responsible for products: their sales then create wealth, which are evaluated according to their contribution to that increase.

From the 1970's to the 1980's, Japanese goods had been accepted for their good quality and reasonable price, and penetrated markets around the world. The Japan Society of Mechanical Engineers (JSME) has also conducted systematic surveys for continuing such success again in the future, and the resultant 11 books were published from Mita Publishing. Nearly one half of them relate to "how to create hits". "How a hit product was born[3]" is one of these, and contains reports written by key people of hit products on their "work, sweat and success" published with their companies'

official permission. Although these techniques are obsolete now, such stories have been rare, and the lessons learnt are still useful.

Table 1 is a very short summary of eight hits[3]. Each title is each product name, and a short description is shown. The rightmost side column shows the awards received, where A is an academic one and I is an industry one, where superscript shows multiple awards and n means many. Each success may be understood by awards, denoted by A's and I's. The keys for success are listed in each case in the following three groups.

*A group: Case 1. Combined-cycle Electric Generator Plant, Case 2. Hot Isostatic Pressing and Case 3. High-Power Laser-Cutter. These are successes from heavy industries. *The basic and fundamental technologies in these areas show a constant rate annual increasing trend.* All companies invest a lot in each research field, such as using excellent people, investing a lot of money and powerful equipment. *During these struggles, a wining developer made most of the opportunities and it became a hit. Most wins are in an oligopoly, protected by their own patents and know-hows; and each company enjoyed a large profit.*

Table 1. Hit products

Group and case		Product [paper] (author and affiliation)	Outline	Award
A Group	1	Combined-cycle Generator Plant[3a] (H. Hiura, Mitsubishi Heavy Industries)	A high-efficiency electric generator using ganged gas turbine and steam turbine, where the steam is heated by turbine exhausted gas. While up grading main technologies every year, they got the first order.	A^n, I^n
	2	Hot Isostatic Pressing[3b] (F. Fukuda, Kobe Steel)	While up grading both technologies of very hot temperature and very high pressure, new ways of purifying and processing of new material, are established, and they became new products of a new technology.	A, I
	3	High Power Laser Cutter[3c] (N. Tabata, Mitsubishi Electric)	High power laser light cutter featuring clear cut-end with high controllability. During yearly enlarging of laser power, silent discharging, which others had abandoned, enabled precision control of laser light for cutting.	A, I^2
B Group	4	Mazda automobile "Familia"[3d] (T. Minami, MAZDA)	MAZDA "Familia" is a Front-Engine Front-Driven car with hatchback door, which created original space. Thus created original space enabled best price car with rich specification.	I^4
	5	*Training Kit* TK-80 and following[3e] (K. Watanabe, NEC)	In 1976, NEC began to sell TK-80 Microcomputer kit, which was welcomed by high-class amateurs. NEC had been top share PC vendor for many years after that.	I^n
C Group	6	Autofocus Single-Lens Reflex camera α[3f] (I. Yoshiyama, Minolta)	The world's first automatic focus S.L.R. camera, which simplified the focusing mechanism by abandoning lens compatibilities. Hereafter, it became the standard architecture of world's SLR camera.	I^n
	7	Miniature Pick-up Lens for CD Player[3g] (Y. Nagaoka, Panasonic)	By using aspheric surfaces of both ends, this is a monolithic structure equivalent to several groups of object lens and condenser lens, and is made with high temperature glass pressing.	I^4
	8	Brassière using Shape Memory Alloy[3h] (H. Nakano, Wacoal)	Past products used metal wire for forming the cups, but during washing, wire extrudes and the shape can be lost. By using S.M.A., it restored the cups when putt on automatically. Millions sold.	

Case 1 had a chance of recording the world level top running product along the trend line. Case 2 created quite new concepts of producing new materials and the forming of material, and *opened its own new product market.* Case 3 created new cutting machines applicable to automobile bodies, and *opened their own new product market.*

*B group: Case 4. Mazda automobile "Familia", and Case 5. *Training Kit* TK-80 and following, are *great hits supported by users. They are planned and managed to fit users.* Case 4: The first key is their excellent technology to implement a Front-engine & Front-drive automobile. It also created new and yet-undiscovered space. The second key is a great product plan to tailor the car to a very cost effective one with attractive specifications. Thus it became a great hit as planned.

Case 5. *Training Kit* TK-80 and following is achieved by Mr. K. Watanabe in NEC, who is respected as the ***"father of PC in Japan"***. He wrote this report. He moved from Semiconductor Development to Sales of LSI microprocessor. They knew that they must nourish these devices, and began to sell the *Training Kit* TK-80 in 1976. The kit consisted of Intel 8080-like MPU by NEC, with a minimum input and output devices. As a computer was still very expensive at that time, it became a great hit among high-class amateurs. For the sales promotion, they built a chain store named "NEC micro computer shop", with a service room named "Bit Inn", where a user could get detailed technical information from NEC. Mr. Watanabe served users there, and then he assigned designers to experience the interaction with users. Designers designed new kits based on that experience. He had persuaded NEC to disclose NEC technical information to the public and then allow third party vendors of TK-80's. These educated and organized users and the market pushed NEC to become the top leading maker in the Japanese PC world with PC 98 families. In parallel with active operation with users, NEC had supported Microsoft as a business partner of hardware manufacture since 1979, and their top position continued also for the Windows PC age.

The lessons of these two examples are that *the users' support is the great prerequisite for business success.* Mr. Watanabe concludes his paper saying *"Although a hit product might be regarded as a happening by chance, actually it is the reward and final result of cascaded cool decisions".*

*C group: Each of following three is inspiring in their respective ways. Case 6. Autofocus Single-Lens Reflex camera α is a dream camera for all camera users. By throwing away the premise of the lens compatibility, autofocus is realized economically, and it becomes a great hit. Case 7. Miniature Pick-up Lens for a photo sensor in a CD Player stands on aspheric lens and high temperature heat pressing of glass lens. As the central research had nourished these two technologies step by step in the past developments, such difficult technologies were completed. Case 8. Brassière using Shape Memory Alloy is achieved by using specially developed SMA and designing the new product line based on design methods using the accumulated data base of human body sizes. The development was made based on previous accumulations of technologies.

Keys for success are summarized as follows:

1. High technology power with good selection ability and the ability to fit the technology to suitable products.
2. Users' support is a great help.

Table 2 is taken from [4], which summarized 161 answers to the questionnaire sent to 1000 major manufacturers in Japan. The questionnaire requested them to point out "up to seven major factors influencing success in a new product development". The right side column shows the percentage selected. Developers can study from this table what and how such factors contribute to success.

Table 2. Factor influencing success of a new product development

	Influencing factors for success	%
1	Unique product	56.60
2	Top management's direction and decision	50.31
3	Clear objective	46.54
4	Corporate R&D capability	46.54
5	Meeting users' needs and demands	45.28
6	Strong leadership supported by long-term vision	44.65
7	Good quality, reliability and inexpensive cost	41.51
8	Originality, cooperation and eagerness of the developer group	36.48
9	Conforming with own production technology and equipment	35.22
10	Predicting future needs	33.33
11	Good timing in entering market	27.67
12	Strong sales route	24.53
13	Concurrent engineering, development productivity and sales	23.90
14	Excellent leader of the developing group	16.35
15	Best fits in with own company' marketing ability	16.35
16	Correct fit with the user	15.72

In Table 1, descriptions are mainly from the developers' inside viewpoint. The influencing factors in Table 2 are wide spread and include various external views. It is usual that a very successful development shows many praiseworthy aspects. It is important that people in a developing group work well, paying sufficient attention to these points.

1.2 Success in emerging market

From the end of the 19th Century to the beginning of the 20th Century, emerging industries appeared as the main players in the business world. Industrialists among them such as Ford (automobile), Carnegie (steel and iron) and Rockefeller (gas) had become the world's richest people.

Around 100 years later, ICT (I: information, C: communication and T: technology) has been changing the world as never before. Bill Gates of Microsoft has caught this opportunity, and is the

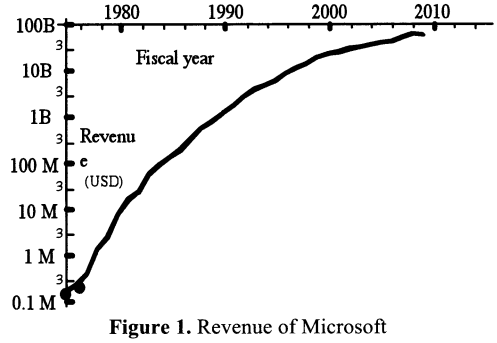


Figure 1. Revenue of Microsoft

top among the world's richest people. Figure 1[original data in 5 and Microsoft's IR reports] shows the growth of the revenue of Microsoft from the start on a logarithmic scale. Bill's strategic management, shown next in a hierarchical manner, was the key to the success. (A strategy is a framework, and when it works it inevitably results in the winning of the game.)

The final object (first level): to become the top of the world's richest people.
Implementation means (second level)
1: To be the worlds top OS for the general-purpose microprocessors
2: To be the world's top for the central tools (i.e. Word etc) on the OS.
3: Make the above two a world level oligopoly.
(Third level) 31: Invest very large money for every enhancement
32: Defeat competitors.

As that time was the growing period of the technology, Gates had to continue his efforts all the time. At present, as technologies as well as society have advanced, this development time has been quite shortened. Good examples are iPod, iPhone and iPad from Apple Computer. Their sales were several million sets in the first year. They are well prepared commercial products like their Macintosh computers.

2. Requirements for top management

2.1 Business viewpoint

In a free and competing market, a manufacturer competes with its product against others. If their product wins and is purchased at a higher price than the competitor's price, the gross profit becomes larger. If the cost is smaller than the competitor's cost, the profit becomes larger. Top management orders people to do their best. For all to do their best, the design they use is especially important.

Figure 2 shows this relationship. The horizontal axis is time elapsed, and the vertical axis is the accumulated amount of money. The accumulated amount of incoming gross profit and net profit are plotted upward, while the accumulated amount of expenditure for design and development is plotted downward. The origin of the graph is the time when the product begins to be sold.

All the curves show simple cases for this explanation. A dotted line, curve S, shows the accumulated amount of sales, and another curve C, also starting from the origin, shows the accumulated amount of cost. Gross profit is defined by the amount of sales - cost, and the vertical distance from the curve S to the curve C is the accumulated amount of gross profit. The accumulated amount of R&D expenditure in curve D begins at the starting time of the development,

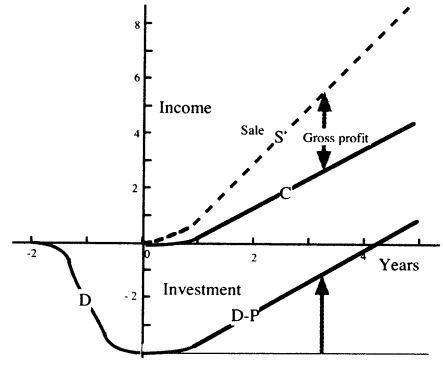


Figure 2. R&D cost vs. profit

goes downward and crosses the vertical axis at the time that all R&D terminates, where the value is the total R&D expenditure used for this product. A fine horizontal line is drawn from this point. The D - P curve started from the crossing shows the recovery of the total R&D expenditure by gross profit, and the crossing with the horizontal axis shows the total R&D expenditure that is recovered. Let us examine several cases:

- ✶ If the total development cost were smaller than usual, the recovery would be finished earlier and the profit would contribute to the company's profit.
- ✶ If the cost were smaller than usual, •the gross profit would increase, or •would be used to decrease the price, in order to get more orders or defeat competitors by decreasing both their sales and their profits.

Thus, improvement of both

• The software size and • the productivity

are the most important issues in any development.

2.2 R&D intensity

R&D intensity is defied by (R&D cost)/(sales). A high R&D intensity shows that a company is at a high technical level ~ new technology oriented. Some example industries are shown in Table3[6], and Table 4[7] shows data from several leading companies.

Excluding Health care[2], Software and Internet is at the top and Chemical and Energy is at the bottom. Rapidly growing industries are at the top. As an industry matures and its leadership in an area has been fixed, eventually it reaches a stable state when the major companies become process industries.

Table 3. R&D intensities[6]

Category	R&D intensity
Health Care	12.8%
Software and Internet	11.4%
Computing and Electronics	7.1%
Aerospace and Defense	4.5%
Auto	4.1%
Industrials	2.0%
Consumer	2.0%
Telecom	1.4%
Chemicals and Energy	0.9%
Others	0.8%

Table 4. R&D Intensity and operating profit

Company	R&D Intensity (%)	Operating profit (%)
Microsoft	13.5	37.6
IBM	5.8	16.1
Oracle	12.2	35.0
Fujitsu	5.0	4.2
SAP	14.2	26.6

Table 4 shows R&D intensities with the gross profit percentage of major companies in software and computers. Microsoft, Oracle and SAP feature the highest R&D intensity group and their main activity is package software suppliers. If a competing company sold a better package software at a lower price than theirs, their business would be undermined. In order to prevent such an eventuality, they have to invest the largest amount of R&D money to upgrade their software and change interfaces in order to bind users. When these are effective, as per user cost, such as is the case with hardware, where it is almost zero; they can enjoy a high profit rate in the oligopoly. As IBM and Fujitsu (although not shown, but NEC and Hitachi are similar

[2] In Health care, as test cost for proving a product harmless is very large and R&D has a high risk, thus R&D intensity is very high.

to them) also manufacture computers, due to their hardware cost, R&D intensities cannot be so high.

Table 5. R&D intensities of Japanese software vendors

Year	'94	'95	'96	'97	'98	'99	'00	'01	'02	'03	'04	'05	'06	'07	'08	'09
R&D I (%)	1.10	1.04	1.33	1.58	1.62	1.72	1.2	1.1	1.0	1.0	1.1	1.02	1.15	0.84	0.47	0.82

The average behavior of ordinary software vendors, however, is different from the above super companies. The Japan Information Technology Services Industry Association (JISA) is an association of major software vendors[3] in Japan. JISA publishes their members' R&D intensity as shown in Table 5[8]. This shows that the R&D intensities had been around 1% for many years. It is said that the situation is also the same in the USA. "Among small vendors, as there are many that feature some technologies, their R&D intensity is rather high. But, in large vendors developing software upon a client's requirement, their R&D intensities are low." As there are no national associations such as JISA in other countries, accurate data is not available. Low R&D intensity means that top management judged that it is more profitable not to invest in technology and R&D for improving productivity and quality. Actually, most advances in this field are those of *product* (e.g. from high-level language to recent cloud technology) not *process*. These have been caused by less severe competition in the market. Most industry people have no experience of research and most academic people have paid no attention to productivity and quality. Thus it is like an engineering desert.

2.3 Software productivity

Let us examine the growth of software productivity from 1980 to 2010. Typical productivity in [9] shows

100 work-month/30K
= 3.3 work-month/K

and that of COCMO data in [10] shows

300 work-month/30K
= 10 work-hours/K.

Japan's recent report[11] shows

10000 work-month/100K
≅ 0.5 work-month/K.

Therefore growth seems to be the order of ten times for thirty years. Growth in hardware logic design, which is very similar to software, seems to be at least around 10^3 times larger than that of software. This is the result of very low improvement activities in software, such as

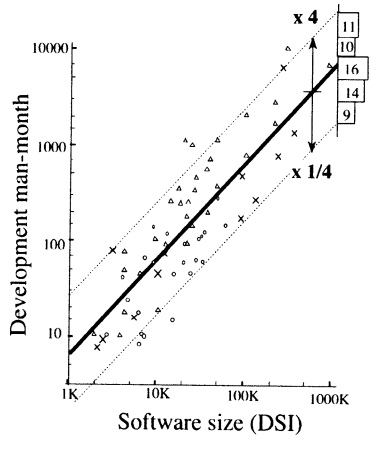


Figure 3. Size vs. man-month

[3] Fujitsu, NEC and Hitachi are the top-computer and the software manufacturers in Japan. They are well known around the world as rare "Software Factories"; their mother bodies do not join JISA.

preferring "no change other than change to improve", as mentioned earlier.

Figure 3[12] shows a software size vs. work-month graph on both logarithmic scales using COCOMO data[10]. The central trend line of the plots shows roughly the relationship of (work-month) ∝ (software size). As [9] had reported it is (work-month) ∝ (software size)$^{1.0}$ as a result of statistical processing of a much large number of data, the exponent 1.0 is used hereafter.

The gradient of the central trend line in Figure 3 shows the average software productivity. Almost all plots distribute in a belt-like zone within two sub-trend lines. Bar graphs, at the top right side of the graph, show the numbers of plots in each of five sub-belts of equal width, and the bar graphs show a bell-shape. Therefore, these show that the distribution is a log normal distribution.

The lower edge of the belt is 1/N times the central average value, and the upper edge of the belt is N times the central average value, and N=4 in the figure. The distribution shows that there is a peak in rather small x value, but it shows a long trail as shown in Figure 4b, in the next page. Therefore the largest and the smallest of the software productivity are as high as $N^2 = 4^2 = 16$ on a linear scale. Statistics say "When *it may be regarded as a multiplicative product of many independent random variables each of which is positive, it shows lognormal distribution.* Originally, characteristics of human works (irrespective of hardware or software) showed similar lognormal distribution. The variation (in this case, difference of productivity) is caused by the difference of the technical capability accumulated[4]. The following is a simple explanation.

The wildly scattered plots of Figure 3 are the results of the many independent variables, as mentioned above. Let suppose that these variables are sorted according to the effectiveness of the productivity. (Refer to the shape of lognormal distribution) There are similar phenomena. Pareto, an Italian economist, found Pareto's principle. It says that a few families (i.e. 20%) own the majority (i.e. 80%) of the wealth of a nation (it is called 20%-80% law). Dr. E. W. Deming was a statistician who worked in the early quality field. He had engaged in statistical quality control. He applied Pareto's principle in quality improvement, and named the approach as Pareto Analysis.

When defects (problems) of quality are sorted as Pareto did, a few vital areas constitute majority of the losses. Take this "few vital areas", and improve the countermeasures to prevent the same defects in the process concerned. As improvements become effective and the quality level is much improved, repeat the same to improve the quality level further. Thus repeating continuously, the quality level improves much better than previously. Dr. Deming's quality education on statistical quality control especially with Pareto's analysis had improved Japanese products quality then they began to penetrate world market from the 1970's to the 1980's. In Japan, it was widely acknowledged that the same arises also in hardware cost and productivity, and now the approach is regarded as a standard procedure for manufacturing in overcoming any problems.

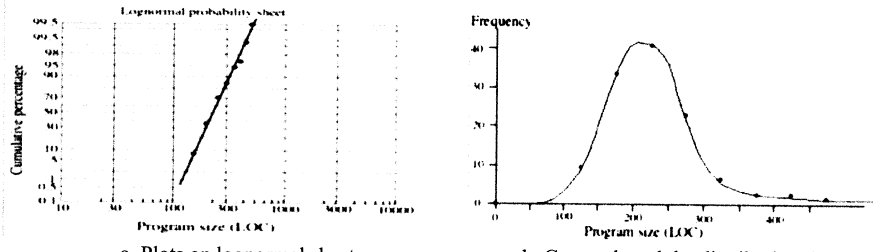
The reason is that all characteristics of what a human being do show lognormal distribution, as easily understood from the definition of the distribution. Therefore any improvement activity requires *firstly quantitative measurements and secondly Pareto's analysis and the following improvements by mostly process changes*, and finally these must be continuously made. This is the reason why there is such large difference in

[4] In Japanese hardware industry, wild variations of lognormal distribution had disappeared before 1960.

productivity between hardware logic and software, although both were at the same level in the 1960's.

2.4 Software size

As for development cost, both productivity and size influence this in the same way. The improvement of both productivity and size may be made successively. But in the design of a mother system, the size, which has effects over a long time period, must be minimized with best practices at the time of the development of the mother system. Here the lognormal nature of software size and various examples of how the size is increased are shown.



a. Plots on lognormal sheet b. Curve plotted the distribution data

Figure 4. DeMarco's experiment

DeMarco made an experiment[13]. He gave a specification to programmers, and people designed the program and debugged. When he examined the distribution of source code size, he was surprise to find that the largest and the smallest ratio were near 10. In Figure 4.a, a linear trend line of plots appears. The section paper is called a lognormal probability sheet, and when plots show a linear trend line, the distribution is lognormal distribution. Figure 4.b shows an approximation curve of the data, which shows an example of lognormal curve.

Each bar graph of Figure 5[14] shows each software size of the central part of three telephone-switching systems of the same functionality.

- Sys Z: A system, developed by programmers "upon requirement".
- System Y: A system, developed jointly by engineers of switching field and programmers.
- Sys X: As designers had previous unsatisfactory experience in software size, they developed a "way to reduce the size structurally".

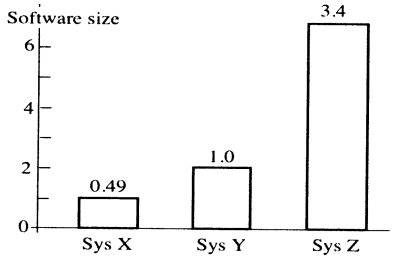


Figure 5. System sizes

Normalizing each size of three systems by taking the medium size Sys Y. This is like a typical software size of 1/N (well studied): 1 (engineering normal): N (by programmers) following lognormal distribution, where N is from 3 to 5.

Let us analyze the inside of the differences between the aforementioned Sys Z and Sys Y. Figure 6[14] shows detailed comparisons of the central parts of operation and maintenance functions of SysY (left side) and Z (right side). The function here is

normalized to the number of commands. The left side and the right side of the vertical axis are Sys Y case and Z case respectively. When compared with both total numbers, they are 65 and 120 respectively. Hatched square boxes show each number of A and B of a category shown by the respective hatch. It is clear that the differences appear in various categories. This shows that the small software size may be achieved by designing in which designers create a design paying much attention to decreasing the size everywhere. In the next 4.1, the inside of the design is explained. In order to achieve a good design, a designer must consider various possible results of the selection at each step of hierarchical decomposition. This is especially important in a high-level design, and if needed a study should be made for making the software size small.

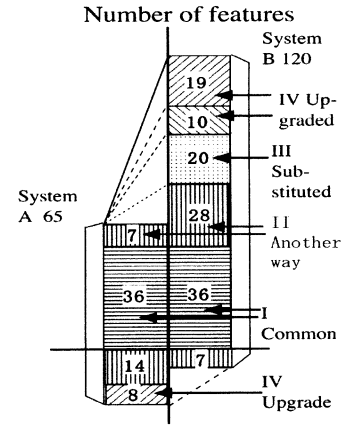


Figure 6. Differences

In this case, there was a conflict between designers who belonged to the company and temporary designers sent from software venders. From the former's viewpoint, every design must be made considering the productivity and other factors. The latter's claim was as follows: they were trained from the dispatching company to "design quickly as is requested by the client. Do not waste time considering how to integrate and make the logic compact and simple. It is easier to amend these later, although tedious. Furthermore, if the client changed their mind, everything collapses." "It is important to predict the expectations of the client for the larger software size in order to get more budget allowance. This is not only for the person involved but also for the developer." This conflict stems from differences of standpoint. It is important to realize these differences and take proper measures to overcome them.

3. The inside of design

3.1 Principle of design

Figure 7[12, 15] shows the design of a clock program. As the authors have reported this already[16], the report is brief in this paper. Hierarchical data flows, on the center left, fix the entire design. Adding both input and output data to the specification "clock", an elementary data flow is created, which is a concept expressed by natural language. This elementary data flow is a parent concept "clock" and is decomposed to detailed children concepts, {"obtain time (Source: S)", "obtain hands (Transform: T)" and "display (Sink: S)"} in a serial manner, by Myers' STS division[17]. A flowchart is also provided to control the execution sequence of children functions. Thus, the initial concept "clock" is hierarchically decomposed to three *serial* concepts, which are mutually independent. *Therefore, this serial decomposition gives a minimum software size.*

In the next step, hierarchical detailing is made of each of the three children concepts. The figure shows the center children concept case of "obtain hands". In this case, both input and output data are decomposed hierarchically to form three parallel children data flows. (This is a case of a typical pattern of Jackson program development[18].)

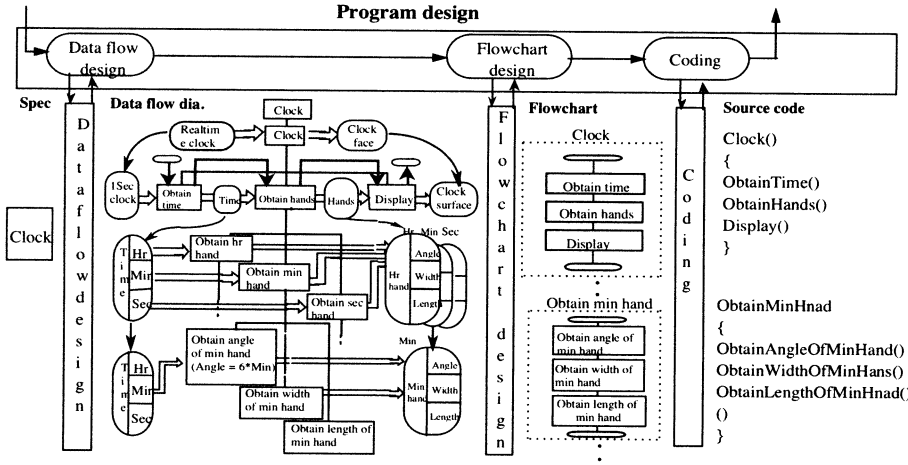


Figure 7. Design of a clock program

Examining these three, although data changes with functions like time/minute/second, each processing is similar to the others. Therefore as each of the three is not totally independent, they form a system of three parallel parts having a similar component, which is used in common. Therefore, by arranging adequate programming, they may be made almost mutually distinct.

These serial and parallel decompositions are the typical basic types of design. These two show, by a careful hierarchical decomposition, that a system may be designed to be the minimum software size.

By checking carefully and rigorously at each step of the decomposition, the average checkout rate of 80%, almost the upper limit, may be achieved. This becomes possible by using accurate and easy to understand descriptions in natural language.

In any decomposition of concept during design, there is a selection of the implementation or decomposition algorithm. It appears in between the parent concept and the children concept. In the bottom stream of design, a parent concept almost specifies the children. But, in the upper stream, there are many candidates for the children. The selection and following decomposition determine the performance, cost and reliability, etc. For a good selection, appropriate research before starting the important decision is necessary. In order to pick up the important or difficult themes the research execution method requires high-level ability with research experience among management.

This design uses the "human intentional activity", achieved by repeating concept decomposition as expressed in natural language. Therefore this may be used in any human intentional activity. An example of the highest management level is shown in Section 2.2 as Bill's strategic management or a pair of "the final object (parent concept of management level) and implementation means (children concept)". Decomposing

hierarchically each of the children, not only the final object is detailed but also the final object gives the guideline to follow Bill's order, until they are reduced to each motion of the lowest level people. This was pointed out by a military thinker Clausewitz[19] and used for planning war.

Thus this concept decomposition is common from the uppermost level concept, then for middle level design and finally the human physical operation level. The repetitive hierarchical decomposition by a human elementary mental operation forms a constant-rate expanding network. Thus, from top management level to bottom physical level this shows the same work-hour and defect number characteristics[20, 21]. This shows that a unified engineering methodology should be considered. *It is not true that a software development process is different from other development processes e.g. hardware).*

A characteristic of this design is that the data flows in Figure 7 shows a design diagram of the logic for a clock. At the most detailed level, a conversion from a natural language parent concept to logic gate expression completes the logic design.

In program design, a pair of natural language expression to corresponding source code is used. What are essential to design are these hierarchical decompositions of the concept involved. The ability for correctness, as many theoretical people insist, is one of the subsequent requirements.

By deconstructing the generation of children concepts, many ways of the automatic design of software become possible[16] and patented[22]. The primary-way (skill based) is to use the actual concept pair as shown in Figure 7. A secondary-way (rule based) is to use frame-type knowledge. A skeleton of children concepts as "data" and "method" is provided to complete descriptions of the children concept. A third-way (knowledge based) is to use the elementary knowledge of a concept dictionary.

After a system has been completed, if it were found that the system had violated some intellectual property rights, it might face legal prohibitions on the use, making, and selling of that method, or a large royalty may have to be paid for the rights, or compensation money for violations. Most of the hardware companies study and try to foresee future problems, start research on important issues, and acquire a patent for some influential key methods or for some narrow paths through which any of them have to pass. A company should invest sufficient funds in researching these issues, and apply for intellectual property rights when an important method is established.

3.2 Detailed process of a commercial product development

An actual product development system is a mixture of pure technical design, as shown in Section 4, and reviewing and production processes by related sections (i.e. Production and Quality Assurance [5] (hereafter QA)), and top management controls all these. Figure 8 is taken from [23] for ease of understanding. This is a prototype flow for a product development process of Sakura Color Products Corp., which is a large manufacturer of office items and stationeries. It aimed at improving the quality of

[5]Quality Assurance (or QA Department in Japan) is the final gate for a product to go out of a company. It is like Inspection in others. Passing the gate, the products are guaranteed by a company to have the quality of that company. Inspection usually means actual works for the assurance. QA belongs and reports to top management. As QA is responsible for the work, QA does everything necessary for the assurance of quality. QA faces customers. QA advises top management not to ship out a product if there are problems. QA plans the quality level, plans how to achieve the goal and executes these plans.

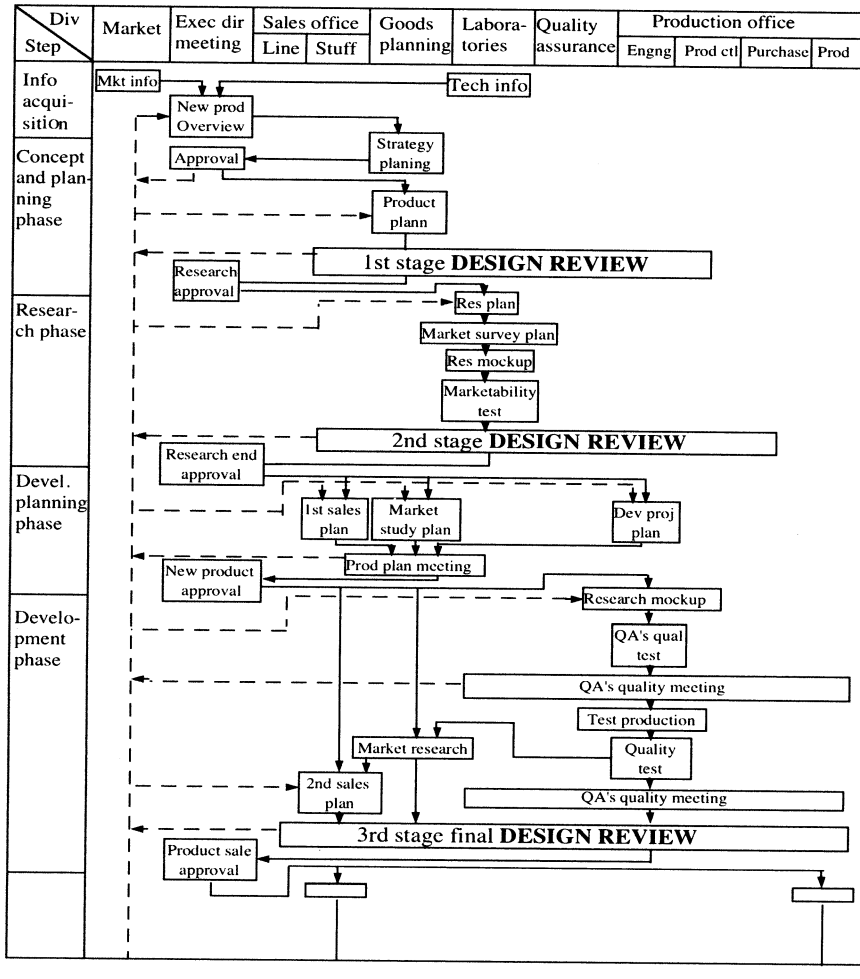


Figure 8. Development process for a commercial product

development management by using the various methods of scientific management. In Figure 8, each box, in the leftmost vertical row, shows the time as the development advances. Each box, in the top most horizontal line, shows each functional name of the organization. In the topmost left of this figure, the starting points are market information from that field, and technical information from laboratories is on the right. They are reported to the Executive Directors Meeting[6] (EDM), which is the place for the top-level management decision by members of the board, where an overview of the new product is discussed, and instructions issued to initiate the Strategic planning of a product. Going downward along the EDM row, a sequence of the management decisions, after the approval of the strategic plan for this development, is shown. Based on the approval of the strategic plan, it is detailed in goods planning up to a product

[6] A Japanese company is like a bottom up system. Before CEO or the president states a final decision, top level director(s) of the board make(s) discussions after receiving an explanation of an issue.

plan, and the first stage Design Review[7] (DR) is made with people from Sales, Laboratories, Quality assurance, Engineering and Production Control in Production.

In Figure 8, the row of the EDM shows major decisions. In the Figure 8, there are second stage DR and third-stage DR. Please follow the description and the flow in the figure.

In order to check out possible problems quickly, DR is made in the early phase of a development. It is rigorous and involves strict reviews and checks by experts of various fields, based on the product documents.

The review results are reported to the EDM, and each director examines them likewise, then the following research-level works start on receipt of the approval. Laboratories present their research plans, and carry out market survey research, laboratory prototype (mockup), and marketability tests. They are reported to the second DR, and then reported to the EDM. As this phase advances, it becomes more and more detailed.

When all these are accepted, the research phase terminates, and it advances to the next phase. The first version sales plan, market study, development of each of the product lines, and evaluation of the overall product plan is made, and they are reported to the EDM. After these are approved, prototypes are prepared and evaluated by Quality Assurance, and a sample production is made and tests by Quality Assurance are repeated. Considering these and the market, the third DR is made and leads to the approval for release of the sales.

As has been described, this phase advances from the research stage to the actual production stage. Various related sections are introduced and work together; the product quality is certified and finally the sales begin. During these processes, top managements also join in the work of checking the products. Requests and advice from related sections are built into the product, and at each phase, evaluations on the market conditions are considered.

In the market of office items and stationery, each product is not so complex, but the number of varieties is large, thus the entire system is crucial. Manufacturers provide various product lines, each including many kinds of products. In the market, they attract users with their various selling points, and expect to be chosen by users for these. Therefore, product strategy and product planning are very important. From the figure, the specialty of the market and Sakura's special strategy are clear to readers. As is shown in this example, all the organization and all the related people see the plans, examine results of the work and the tests for understanding, and through questions and answers all the people involved gain the same information, and perform their respective works synchronously, and thus all the sections and people involved share the work of a development, and thus the products are shipped out.

It is an organic process. In Figure 7, the detailing goes from the top toward the bottom. In Figure 8, the detailing goes in the organization. This is just an example. Each company has their own development process for a commercial product, and the system is the key factor of the company. "Software" is just an implementation means. In a company, many people with different skill/knowledge/technology work together

---

[7] In the original DR, people prepared explanation documents such as system specification and maintenance plans, and experts such as sales, systems planning and maintenance read and prepares advice, evaluation or comments before meeting, and the DR was then held. It aimed to supplement where the designers' knowledge was insufficient. But, DR is executed in various ways in Japan. In the Sakura case, emphasis seems to be put on experts' (from every fields) views including various data and discussion freely held during reviews, in order to pick up and solve various problems and guide people toward the final target.

and thus integrated power forms a commercial product. The business model, which postulates that software be produced based on market requirements, does not fit here. The business model of Figure 8 necessitates the software should be distributed all through the various stages. The present duties of all the sections must be extended so as to include some of the software. It is probable that most companies have been approaching this new style.

3.3 People

It is generally accepted that a business's success depends on people. An important point is how to extend education after university graduation. In Japan, a 4th year student graduates from universities in March. On the 1st of April, when a new fiscal year starts, every company has a welcome ceremony for their new employees. The president of each company welcomes them and encourages them to be good workers. As the president's address includes management philosophies introducing company mottos as well as the corporate direction for the new fiscal year, it is impressed on new graduates what is the ethos of the company.

After new employees are distributed to each factory or laboratory, the respective education programs start as On the Job Training (OJT). The following is an example. After the guidance in the initial training, the new graduates are organized to form small teams. People are dispatched to Akihabara, and each team buys some kind of an electronic product (i.e. telephone etc.). On returning back to the company, each team examines the functionality of the product, disassembles the set, examines the inside and each writes an analysis report including things such as the component list as evidence. Teams with the same kind of the set (e.g. telephone) gather together and compare their analyses; they discuss which is better in appearance, price and cost, functionality, easiness of operation, etc., and finish with cost studies. Then people discuss eagerly how to make their own set, and try to exceed others. Finally all the members get together and each makes a group presentation. Thus, *"Market competition and Cost"* are strongly impressed on new people.

Another important part of early OJT is Industrial Engineering (IE) in production. New workers see a distribution graph of work-hours per job. As the work procedures are standardized and more strongly constrained, the distribution becomes sharper. They also learn the history of how the average work-hour per task (namely the productivity) has been reduced successively over decades. They learn how *people's effort and training, tools and aids, automation system, together with the progress of technology* (i.e. mechanics, electrics, electronics and information) have achieved that progress. The instructor emphasizes that *this evolution will continue also in the future*. Then, there is a short lesson of IE[8][24] including Quality Control[25].

Various experiments, assigned by the leading engineers, are another good opportunity to educate new workers. A half-day after the start of the experiment, a new graduate came back and told the engineer. "The data are unreasonable values. It is a *mystery!*" Both then went to the laboratory, and the engineer postulates a hypothesis, takes data, then extracts a conclusion, and then repeating the same, and finally the cause of the instability is found and corrected. The client explains, "Due to the too long

[8] IE includes all about Scientific Production Control (quality issues are included), developed by pioneers such as F. W. Taylor since the beginning of 20th century and enabled the present prosperity of industries by rational, quantitative and scientific approaches.

leads, the circuit falls in a *parasitic oscillation of very high frequency*, which results in unreasonable data." The engineer emphasizes following;

> "As you feared, you misunderstood the happening as a *"mystery"*. You lost your scientific commonsense and so the problem has become a mystery. Do not be afraid. Look at the item closely, and treat the problem scientifically".
>
> "Measure quantitatively, and plot the results to form a graph. From the trend appearing in the graph, some idea would be obtained by abstraction. Thus repeating, it becomes clearer, then finally the problem, that you called mystery, will melt away".
>
> *"If it is still unclear, it must be a limitation of our company or the present technology and science. Stop at that point.* Think about another way to achieve the object.

Besides these, various basic principles are taught. Some important issues on the "structure and rules of society" are as follows:

- Hierarchical system: formal structure of an organization is repetitively hierarchical[9]. It also implies the hierarchical "authority and responsibility" system.
- Profit center: This has all the authority and responsibility for a product. (If Sales is the center, they direct Design as well as Production. If Design is the center responsible for a product, Design orders the production quantity and the dispatch date with cost attached to Production.)
- Top down principle: A higher-level person has authority over subordinates but has to take responsibility for what the subordinates do. Later, that person trains the subordinates how to avoid making a mistake.
- Responsibility: It is a golden rule that in a person's work results there should be no defects. For this, one has to check for faults after a work is complete. If a defect is found later, the person responsible has to amend it on that person's time. (This achieves the low defect intensity of hardware production.) If a defect is found in the output of a design group, the loss is charged to the group. (Therefore tester team's salaries are charged to the design group. The design group's manager must make efforts to decrease remaining defects[10]. This has been an incentive to introduce simulation in a design.)
- Regular work: On receiving an order, one has to do the work by oneself as instructed, or to decompose the order to the means to attain it, and then execute detailed orders to others. On receiving a design document, a person has to transform it in a strictly manner. If this includes some problem, it should be discussed with the person who prepared it. One should never distort nor change anything without the permission of the previous designer responsible for that work. Generally, software people are not aware of this authority and responsibility rule.

---

[9] Irrespective of whether software people agree or not, this is a rule in the society. In the early 20th Century, there was a dispute, and after field experiments, the hierarchical structure was found to be better. Since then, this has become a rule throughout the industry.

[10] If such efforts were not made, designers would be free of the burden of improving their quality, a tester team could have higher salaries, the users would be annoyed by the poor quality and unwilling to buy the products, and the company would get a bad reputation.

4. Discussion

Due to progress in the speed of communication and transportation, the world has become one global market. In such a large, free and competing market, there are severe fights between commercial products of each category in each field. If the total of the number of components, volume or weight is larger than those of competitors, the cost per unit is high. If technology is more advanced than those of competitors, the cost per unit becomes more economical. As a result, it is said that only companies in the world's top levels can survive. If a company survives through these wars, the company can enjoy a large profit in their oligopoly.

As has been shown in this paper, in *the current software engineering and the industries*, there is no such severe competition. It seems to be like an "eyewall" in a strong tropical cyclone. Just outside of this low competition society of software, there have been severe wars in OS and database, where Microsoft in OS and Oracle in database have been the winners enjoying an oligopoly. As has been described in Section 2.2, they have been investing a lot of money for R&D to keep their top position in the industry. Thus, *the current software engineering and industries developing that engineering* are a special case.

Since the 1990's, banks had been integrated to form a few mega-banks due to the large investment in banking software systems. Recently, the investment for IT software systems has become a heavy burden even for medium level companies. The percentage of packaged IT software has been increasing in rather small to medium companies. *SaaS (Software as a Service)* and *cloud technology* will economize the burden even for large companies. These mean that the percentage of custom-made software will be deceased in future. It is still in the pioneering stage and a new kind of system, yet it is expensive and custom-made (e.g. enterprise system). As the technology matures, it becomes a commercial product in a free and competing market, and finally it ends as one commodity. Thus, the need for *the current software engineering and industries developing that engineering* will be decreased. A clear symptom of the beginning of the decline is the fact that the decrease of the number of applicants to the computer program related departments. They know that around ten times large numbers of students are learning Computer Science in emerging countries, and many software vendors are outsourcing offshore software developments. Students know clear evidence that software development is labor intensive. From their viewpoint, it is low-tech industry.

5. Conclusion

Current software engineering postulates that software is developed upon the client's requirement. A software vender details the requirement and develops the custom-made system. This is one business model.

There is another business model also for software. A company (it may be a person) does business in a free and competing market. In order to be successful in that business, it develops their commercial product or service. The system might be purely software, software with hardware or purely hardware. As has been discussed in Section 3.1,

human intentional activity, from management activity, design and physical works, is performed by repetitive concept decomposition. Therefore hardware and software are the implementation means for achieving the final object of an organization. Such has been the business model since modern industrial society started at the end of the 19th century. Major software companies and various embedded system vendors have been working in this business model. The authors claim that this business model should be the main model.

It is an important mission of engineering to pave a road to enable people, companies and industries achieve success. Considering this, the authors explained major patterns of the activities involved in some detail. The campaign of "hits" by JSME, introduced in Section 1.1, is one good example.

In Section 1, various hits are described. Through it, various keys for success are shown. Marketing and Product Planning are two early fields. As various scientific studies have been made and wider routes for success have been known, it has been emphasized that this is a search for needs rather than acting after the needs have become apparent. After the main routes have been become clearer, the remaining and the largest problems are how to *create a hit.*

In Section 2, starting from management requirements, various business viewpoints were introduced, and the key factors such as productivity and software size were discussed in detail. Through these, quantitative explanations and graphs were shown as an indispensable means for engineering. By virtue of these, one can understand a situation precisely, and various cause and effect relationships are known. By accumulating knowledge or stacking up technologies, variations may be decreased, and various characteristics begin to advance. These arise in all engineering. Unfortunately, some are still wandering in a mist of unknowns.

In Section 3, more detail of the foundation of design was discussed and detailed application style was also shown. Major activities for a development are based on "human intentional activity"; therefore, any developments are of the same structure. That is, rules and processes for software version are similar to hardware versions. (Please note that special rules and processes specific to software are few and in a limited area.) It is necessary for software people to know them to keep following them.

In thus doing, people can save energy, and focus on each creation, aim at success for their organization and contribute to the benefit of society and the world.

Acknowledgements

The authors wish to express thanks to Dr. Hassan Abolhassani and other students as well as Dr. B.H. Far, formerly Associate Professor in the laboratory, and participants in the Software Creation Project in Saitama University for their contribution of this study. They are also thankful to Hitachi, Ltd. and other organizations for their sponsorship to the project. They are thankful to Mr. Daniel Horgan for his careful correction of their English. Koono expresses his gratitude to many people for their advice, co-operation and support. Finally he thanks the late Emeritus Prof. H. Inose, of the University of Tokyo, for opening wider views and giving people encouragement to break through the cloud of unknowns!

References

[1] Z. Koono and H. Chen, Top Management Conducts an Enterprise System Development, *Frontiers in Artificial Intelligence and Applications, SoMeT_09* (2009), Vol.199, 389-395.

[2] Y. Naya Ed, *Research and Development and TQC-Strategy and Penetration to Industries*, Japan Standards Association, 1990, (in Japanese).

[3] Eds. Japan Society of Mechanical Engineers, *Report from Forefront of Development: How a Hit Product was Born, Mita Publishing* (1987), Following a to h are papers in this book, (in Japanese).
a: H. Hiura, *Development of a Large Capacity Combined-Cycle Electric Power Generator Plant,* 217-230.
b: T. Fukuda, *Development of Hot Isostatic Pressing (HIP) Product,* 37-80.
c: N. Tabata, *Development of High Power Laser by Non-voiced Discharge,* 7-35.
d: T. Minami, *Development of Familia Series Automobile,* 143-164.
e: K. Watanabe, *A Personal Computer Series: From the Development up to Now,* 105-139.
f: I. Yoshiyama, *A Survival Story of Auto Focus Camera α,* 81-104.
g: Y. Nagaoka, *Development of Ultra Precision Non-aspheric Lens Product,* 181-213.
h: H. Nakano, *Nice Fitting with Relaxed Mind in High-tech Age,* 165-180.

[4] F. Kurokawa, *Success Factors for Various Types and the Case Study*, Dokkyo Studies in Data Processing and Computer Science No. 21, Center for Data Processing and Computer Science, 2010, (in Japanese).

[5] editors@thocp.net, *Microsoft Company 15 September 1975-1990 and 1991-2005.* http://www.thocp.net/companies/microsoft/microsoft_company.htm and http://www.thocp.net/companies/microsoft/microsoft_company_part2.htm

[6] B. Jaruzelski and K. Dehoff, *Profits Down, Spending Steadily: The Global Innovation 1000*, pp. 6, preprint, strategy + business, issue 57, Winter, 2009.

[7] Department for innovation, Universities and Skills, *Software and Computer Service, Sector Summary*, The 2008 R&D Scoreboard, 2008.

[8] Japan Information Technology Services Industry Association (JISA) Ed., *R&D Intensity*, Annual Basic Statistics of JISA for every year, (in Japanese).

[9] R. Nelson, *Software Data Collection and Analysis,* Rome Air Development Center, Air Force Systems Command, Griffiss Air Force Base, 1978 (Western Michigan University Library has one partial copy).

[10] B. W. Boehm, *Software Engineering Economics*, Prentice hall PTR, New Jersey, 1981.

[11] Software Engineering Center, *White Paper on Data of Software Development 2007*, Information-Technology Promotion Agency, 2007, (In Japanese).

[12] Z. Koono, K. Ashihara and M. Soga, *Structural way of Thinking as Applied to Development,* IEEE COMSOC/IEICE GLOBECOM '87 (1987), 26.6.1-6.

[13] T. DeMarco and T. Leister, Software development; State of the art vs. state of the practice, *Proc of 11th ICSE* (1989), 271-275.

[14] Z. Koono, T. Kondo, M. Igari, and K. Ohtsubo, Structural way of thinking as applied to good design (Part 1. Software size), *IEEE Global Telecommunications Conf. 1991* (1991), 24.3.1-8.

[15] Z. Koono, H. Chen and B.H. Far, Expert's Knowledge Structure Explains Software Engineering, *Proc. of Joint Conference on Knowledge-Based Software Engineering 1996* (1996), 193－197.

[16] Z. Koono, H. Abolhassani and H. Chen, A New Way of Automatic Design of Software (Simulating Human Intentional Activity), *Proc. of SOMET06* (2006), 407-420.

[17] G. J. Myers, *Composite/Structured Design*, Van Nostrand Reinhold, 1978.

[18] M. A. Jackson, *Principles of Program Design*, Academic Press, 1975.

[19] Carl von Causewitz, *Vom Kriege,* 1832.

[20, 21] Z. Koono. H. Chen and H. Abolhassani, An Introduction to the Quantitative, Rational and Scientific Process of Software Development (Part 1 and 2), *Proc of SOMET 07* (2007), 361-371 and 372-390.

[22] Z. Koono, H. Abolhassani and H. Chen, *Japanese Patent application, International number W02002/097727,* International publication date2002.12.5, Priority date 2001. 5. 28 ,
*Chinese Patent ZL 02 8 10859.0, July 26, 2006.
*US Patent US 7,480,642 B2, Jan. 20, 2009.

[23] Ed. Y. Naya, *Research and Development with Total Quality Control: Strategy and Penetration to Companies*, Japanese Standards Association, 1990.

[24] IE Textbook Committee of JUSE, *Basic Textbook on Industrial Engineering*, JUSE press, 2000, (in Japanese).

[25] K. Hosotani, *Seven Tools for Quality Control (New version)*, JUSE press, 2006, (in Japanese).