NAIST-IS-DT9761016

Doctor's Thesis

Model Based Project Management for Software Development

Noriko Hanakawa

February 1, 2000

Department of Information Systems
Graduate School of Information Science
Nara Institute of Science and Technology

Doctor's Thesis
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
DOCTOR of ENGINEERING

Noriko Hanakawa

Thesis committee: Katsuro Inoue, Professor
Koichi Nishitani, Professor
Ken-ichi Matsumoto, Associate Professor

Model Based Project Management for Software Development*

Noriko Hanakawa

Abstract

This thesis proposes a project management method based on a new process model and a new simulation model. The proposed method is adaptable to paradigm shifts in software development. In the proposed method, a software development process model is generated from software development methodology. The development methodology defines relationships between activities and products. Development phases and the sequence of the phases of the process model are identified on the basis of the definitions of the methodology. Moreover, the process model includes milestones that are established just behind the identified phases. Even if a conventional methodology is shifted to a new one, a development process model with relevant milestones against the new methodology can be generated. In addition, more accurate development periods can be estimated using a new simulation model in the project management method, when new development environments and techniques are applied to projects. The simulation model considers productivity's variation that is caused by developers' learning effect. Productivity increases as an activity progresses because the developers become more familiar with the new environments and technologies over time. The thesis also presents a new prototype tool based on the proposed project management method. The established plans in the prototype can define a more exact sequence of development phases and milestones, moreover, the plans show development periods which are influenced by variation of productivity by developers' learning.

*Doctor's Thesis, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DT9761016, February 7, 2000.

Chapter 1 clarifies problems of project management for software development. Especially, adaptability of conventional development process to new development methodology such as object-oriented development methodology is discussed, difficulty of estimation of development periods based on productivity's variation which is caused by developers' learning is shown. Chapter 2 shows conventional project management methods, especially, generation of process and estimation of development periods. Chapter 3 describes a method for generating correct development process model as a framework. The framework gives us a guideline for generating process model with relevant milestones for object-oriented projects. Chapter 4 proposes a method for estimating correct periods as a simulation model. The model can simulate progress and periods based on variation of productivity by developers' learning. In addition, to apply the model to industry, the model is customized for easy use, the model is evaluated for assurance of the reliability of the simulation results. Chapter 5 presents a prototype tool based on the proposed method and model. Chapter 6 concludes this thesis with a summary and future topics.

Keywords:

Software development process, simulation, learning model, project management, virtual project

List of Major Publications

1. Noriko Hanakawa, Hajimu Iida, Ken-ichi Matumoto, Koji Torii: "A Model for Managing Projects in the Various Software Development Processes," Japan Society for Software Science Technology, SP-96-10,pp.59-66, Mar. 1996.

2. Noriko Hanakawa, Hajimu Iida, Ken-ichi Matumoto, Koji Torii: "A framework of generating software process including milestones for object-oriented development method," Information of Processing Society of Japan, SE-96-110, pp.55-62, Jul. 1996.

3. Noriko Hanakawa, Hajimu Iida, Ken-ichi Matumoto, Koji Torii: "A framework of Generating Software Process including Milestones for Object-Oriented Development Methods," Proceedings of Asia-Pacific Software Engineering Conference, pp.120-130, Dec. 1996.

4. Shuuji Morisaki,Noriko Hanakawa, Ken-ichi Matumoto, Koji Torii: "A learning curve based simulation model for software development," Proceedings of Foundation of Software Engineering 97, no.19, pp.75-82, Dec. 1997.

5. Noriko Hanakawa, Shuji Morisaki, Ken-ichi Matumoto: "A learning curve based simulation model for software development," Proceedings of 20th International Conference of Software Engineering Vol.1, pp.350-359, Apr. 1998.

6. Noriko Hanakawa: "A simulation model for software development based on developer's learning curve," Proceedings of 20th International Conference of Software Engineering Vol. 2, pp.168-169, Apr. 1998.

7. Noriko Hanakawa, Ken-ichi Matumoto, Koji Torii: "Evaluation of a learning curve based simulation model for software development," the Institute of Electronics, Information and Communication Engineers, Information and Systems Society, KBSE98-28, pp.49-55, Nov. 1998

8. Noriko Hanakawa, Ken-ichi Matumoto, Koji Torii: "Application of learning curve based simulation model for software development to industry,"

Proceedings of the 11th International Conference on Software Engineering and Knowledge Engineering, pp.283–289, Jun. 1999.

9. Noriko Hanakawa, Hajimu Iida, Ken-ichi Matumoto, Koji Torii: "Generation of object-oriented software process using milestones," International Journal of Software Engineering and Knowledge Engineering, Vol.9, No.4, pp.445-466, Aug. 1999.

10. Noriko Hanakawa, Ken-ichi Matumoto, Katsuro Inoue, Koji Torii: "A software development progress simulator based on developer's learning effects,"(Submitted to Transactions of the Institute of Electronics, Information and Communication Engineers) (in Japanese).

Contents

1. Introduction 1

1.1 Software development paradigm shifts 1

1.2 Software development project management 2

1.3 Problems of planning by the paradigm shifts 3

1.4 Outline of the thesis 6

2. Project Management 8

2.1 Software development process 8

2.1.1 Method and process 8

2.1.2 Project milestones 10

2.1.3 Problems of control in object-oriented development projects 11

2.2 Resource estimation 12

2.2.1 Estimating methods 12

2.2.2 Problems of estimating models 13

3. Generation of Software Development Process 15

3.1 Proposed framework 15

3.2 Application 23

3.3 Conclusion 34

4. Estimating Model based on developer's learning 35

4.1 The proposed model 35

4.1.1 The three submodels 35

4.1.2 Simulation using the proposed model 39

4.2 Case study 42

4.3 Customization and evaluation 48

4.3.1 Approach 48

4.3.2 Customization 49

4.3.3 Evaluation 53

4.4 Application 58

4.4.1 A Simulator 58

4.4.2 Case study 58

4.5 Conclusion 63

5. The Project Planning Prototype **65**
5.1 The four components 65
5.2 A feature of the prototype 68
5.3 Conclusion 70

6. Conclusion and Future Research **71**
6.1 Summary of major results 71
6.2 Future works 72

Acknowledgements **75**

References **77**

Appendix **81**

A. Managers' responses of quetions **81**

List of Figures

1 Relationship among a method, a process, and constraints 9
2 The object-oriented development process with traditional phases and milestones 11
3 The proposed framework of generating software development process with milestones 16
4 *DFD* of the *Structured Development Method* 17
5 *ALGORITHM 1*: Algorithm for phase identification along with its input and output data structures 19
6 Phases of the *Structured Development Method* identified by *ALGORITHM 1* 20
7 The Phase Sequence of the *Structured Development Method* identified by *ALGORITHM1* 20
8 *ALGORITHM 2*: Algorithm for baseline product identification along with its input and output data structures 21
9 A unit process model of the *phase and milestone based management process* 22
10 A model of the *phase and milestone based management process* for the *Structured Development Method* 23
11 The object diagram of *phase and milestone based management process* 24
12 The event trace diagram of *the phase and milestone based management process* 25
13 The state diagram of *the phase and milestone based management process* 26
14 The data flow diagram of *the phase and milestone based management process* 27
15 Phases and baseline products of *OMT* identified by *ALGORITHM1* and *ALGORITHM2* 29
16 The phase sequence of *OMT* identified by *ALGORITHM 1* 29
17 A general model of *the phase and milestone based management process* for *OMT* 30
18 Phase of the modified *OMT* 30

19 The phase sequence of the modified *OMT* 30
20 Phases and baseline products of *OOSE* identified by *ALGORITHM1,2* 31
21 The phase sequence of *OOSE* identified by *ALGORITHM1* 32
22 A general model of the phase and milestone based management process for *OOSE* 33
23 Outline of the proposed model 36
24 Activity model 37
25 Productivity model 37
26 Knowledge model 38
27 Learning curve in the simulation 42
28 Result of Case1 44
29 Result of Case2 45
30 Result of Case3 46
31 Result of Case4 47
32 Managers' estimation 55
33 Results of simulation with managers' responses 55
34 Result of evaluation 57
35 Simulator 59
36 Result of Case1 60
37 Result of Case2 62
38 The outline of Project planning prototype 66
39 Assignment of resources 67
40 Schedule for software development 69
41 Re-schedule for software development 69

List of Tables

1 The default values of the parameters 42
2 Estimation and results of simulation with "all revised constants = 1" 51
3 Results of tuning for revised constants of extended equations in step3 52

4 Virtual Project No.1 54
5 Responses about Virtual Project No.1 56
6 Results of estimation 58
7 The values of the engineer rank 68

1. Introduction

1.1 Software development paradigm shifts

In the fast changing Information Technology society of today there have been various paradigm shifts almost daily in software development projects. For example, a development methodology has shifted from a conventional structured development method to an object-oriented development method. According to the changes in the development methodologies, developers have to execute different activities(e.g., discovering objects rather than creating bubbles in a data flow diagram) and, produce different products(e.g., class diagrams rather than a moduler based design structure) and, design by different development perspectives (e.g., object-oriented rather than function-oriented).

On the other hand, software development environments are shifting from "Text editor & compiler" to CASE tools which support many useful functions (e.g., automatic correction of programs during editing). In different development environments, developers need to change their development style. The development style means: "*a kind of development in which forms are not defined in development methodologies*". For example, about twenty years ago, when there was only one computer shared among members of a project, developers of the project had to produce complete products(e.g., programs) on their desks using paper and pencil. The developers then asked a typist to punch many cards based on the programs which the developers believed where perfect on paper. The punched cards where then inputted to the computer by a card reader. After that, at last, the developers could compile their programs and test the programs. Because the developers required so much effort until the testing of the programs, the developers' test chances were two or three times a day. Grammatical errors within the programs could not be found until after compiling the programs. However, now, developers develop products(programs) based upon the CASE tool approach which supports many useful functions. The developers do not need to produce perfect products on their desks, because the CASE tool can correct many errors such as grammatical errors in the programming language automatically, e.g., if a developer edits " fi " in a program on the tool, the tool corrects from "fi" to "if" automatically. In addition, even if the programs are not complete, the developers

will be able to test parts of the programs repeatedly with the debugging system on the CASE tool.

1.2 Software development project management

Management activities of software development projects are as important as the technical activities[34]. Some typical reasons people give when software is not delivered on schedule, are exemplified by the following[31]:

- the programmers did not tell the truth about the actual status of their code;
- management grossly underestimated the time needed to complete project;
- management did not allow sufficient time to carefully plan the project;
- the real status of the project was never made clear;
- the programmers' productivity turned out to be considerably lower than expected;
- the customer did not know what he wanted.

In addition, software development project management activities are classified into these categories[32]:

- Planning: Predetermining a course of action for accomplishing organizational objective;
- Organization: Arranging the relationships among work units for accomplishment of objective and the granting of responsibility and authority to obtain those objectives;
- Staffing: Selecting and training people for positions in the organization;
- Leading: Creating an atmosphere that will assist and motivate people to achieve desired end results;
- Controlling: Measuring and correcting performance of activities toward objectives according to plan.

Especially, the planning of a project may be the most important management activity[1]. Without a proper plan, no real monitoring or controlling of the project is possible. Many failures caused by mismanagement can be attributed to lack of proper planning[6]. Under a disagreeable project plan, even if all developers of a project are mature, the project will fall into utter confusion.

1.3 Problems of planning by the paradigm shifts

Activities of planning are exemplified by the following [32]:

- The tasks to be performed by the software development staff in order to develop and deliver the final software product. This endeavor requires partitioning of the project activities into small, well-specified work packages. We call the work package as "phase".
- The cost and resources necessary to accomplish the overall project and each project phase.
- The development process to specify the periods of the phases, dependencies among phases, establishes project milestones.

A main problem of planning a software development project is indifference to paradigm shifts within the field of software development. The shifts of software development mean "*changes of development methodologies and development styles*", as mentioned in section 1.1. If a conventional planning technique does not adapt to the new development methodology and style, the project will fall into utter confusion using a disagreeable project plan. There are two main reasons for the confusion. The following paragraphs show these reasons.

(1) Not adaptive development process to new methodologies

The first reason is that managers can not comprehend and control their projects, because they tend to use incorrect processes of software development. Especially, the traditional phases(software requirement specification, preliminary design, detailed design, coding, unit test and integrated test) which are defined in the conventional planning technique are not adaptable to new development methodologies such as object-oriented development methodologies. Yourdon notes that in the past few years many software managers have said, "Look, I

don't care which object-oriented methodology we use -It doesn't matter whether it's Coad-Yourdon, Booch, Rumbaugh, Martin-Odell, Shlaer-Mellor, Jacobson, or someone else. What I want to know is whether the project is under control and whether my people are doing the right thing at the right time [35]." The behavior of the project team members is the antithesis of what the manager wants to see: when the manager asks the team members what they are doing, the answer is, "A little bit of everything: some analysis, some design, a little coding, and a little testing." The manager naturally worries that such behavior may lead to the worst characteristic of a software project [35].

In order to comprehend and control the development progress, a phased development is typically employed by using a number of clearly identifiable milestones which are established between the start and end of the project [34]. In software development a typical milestone is the date on which coding is completed and the codes pass reviews. Once a work product has been reviewed and agreed upon, it becomes a baseline product and can be changed only through formal procedures [28]. A good milestone is characterized by finished documentation, for example, "High-level design complete" or "Test plan formulated." A poor milestone, in contrast, is "Coding 80% complete," since there is no objective way of telling if coding is 80% complete [30]. A good reason for the widespread adoption of the 'waterfall' model of the software process is that it allows for the straightforward definition of milestones throughout the course of a project. In alternative approaches, such as exploratory programming, milestone definition is a more difficult and a less certain process [30]. The project manager's "road map" is likely to be different with different phases, different milestones, and different checkpoints [35].

This thesis proposes a new framework which gives us a guideline for generating a software development process with relevant milestones for object-oriented development methods. The framework provides algorithms for identifying development phases and baseline products based on relationships among activities and products of the development method. The baseline product is a work product to be checked at a milestone, and is referred to by the activity of subsequent phases. In addition, the framework defines a software development process to manage development progress in which milestones are established at the end of each phase

in order to check the baseline product and establish goals of the following phase.

(2) Not accurately predicted periods

The second reason is that the predicted periods to execute phases are not accurate. Usually, managers tend to estimate shorter periods[34]. Even if the software development process is correct, developers will not be able to develop complete products during too short a period. At the end stage of the project, the incomplete activities by reason of too short periods make the final product's quality low. Especially, it is important for managers to estimate the periods with productivity's variation which occurs by developers' learning, because it takes much time to be familiar with a new software paradigm such as CASE tools, OSs, hardwares and object-oriented development methods.

In recent years, many software developers are under the pressure of keeping update with new development environments, and chasing after the latest version for software packages. As sufficient training for a particular development environment such as a CASE tool, before its initial application, becomes costly and impractical, the developers have to use the new environment without sufficient previous training for the environment.

In such a situation, the productivity of the software may increase as the development activity progresses. In the early stage of the project, the productivity is low, since the developers need to be familiar with the new environment and they may have to re-execute a part, or some parts, of their activities for a shortage of the required skills[4][13][25]. The need of such learning processes is reduced as their activities progress, as well as the need of re-execution of a part, or some parts, of the activity. That's why the productivity becomes higher and higher. The variations of the productivity depend on the developers' knowledge and experience and their learning curves.

The variation of the productivity in the project makes it complicated to estimate and/or predict periods of phases in projects. That is, we have to take into account the effect of developer's learning during the project in estimating and/or predicting the development periods and the total work efforts. However, the conventional estimation technique for software development claims that there is no variation in the productivity of the execution of an activity[14] [17] [19].

This thesis also proposes a new software development simulation model which considers the variance of the developer's productivity during software development. The proposed model assumes that the developer's knowledge of an activity increases by executing the activity, and the productivity in executing an activity and the quantity of gain to the developer's knowledge are determined based on the relationship between the levels of the developer's knowledge and the required knowledge to execute the activity. Especially, to compute the productivity, a cumulative normal model *Ogive model* is used, thus the productivity drastically changes even if the levels of two kinds of knowledge are almost the same. By plotting the level of the developer's knowledge in time sequence, we can obtain the developer's learning curve during the execution of the activity. In addition, to apply the simulation model to industry, the model is customized and evaluated. As a result, project managers can predict accurate periods of phases in their projects with only answering some questions about the projects.

1.4 Outline of the thesis

Chapter 2 shows conventional methods of project management, in section 2.1, conventional methods of generating a process for software development is shown and presents problems in progress control of an object-oriented software project. Section2.2 describes conventional methods for estimating resources and presents problems of estimating. In chapter 3, a new method for generating processes is introduced. Section 3.1 describes algorithms for identifying software development phases and baseline products, and a software process model for managing development progress with milestones. Section 3.2 presents applications of the proposed framework to two well-known object-oriented development methods: *OMT* [27] and *OOSE* [15], and discusses adjustment of the length and/or the number of phases. Section 3.3 summarizes the ideas discussed in this framework. Chapter 4 shows a new model for estimating periods of development projects. Section 4.1 describes the proposed model with its assumptions and parameters, and how to estimate the development time and the total work effort. Section 4.2 presents applications of the proposed model to four cases in which the characteristics of activities and developers may influence the software productivity and the development progress. The proposed model's characteristics and implications will be

discussed based upon the case study. Section 4.3 presents the customization and the evaluation of the model for applying the model to industry. Section 4.4 shows a new simulator based on the model. Applications of the simulator are presented in six cases in which the characteristics of activities and developers may influence the software productivity and the development progress. The simulation model's usefulness and characteristics will be discussed based on the cases. In section 4.5, the summary of the model is shown. Chapter 5 describes a prototype of project planning based on these proposed methods. Chapter 6 summarizes the finding and presents some possible areas for future researches.

2. Project Management

A management model comes from management science[18]. It is universal model in the sense that:

- Management performs the same functions(planning, organizing, staffing, leading, and controlling) regardless of position in the organization or the enterprise managed.;
- Management functions are characteristic duties of managers; management practices, method, activities, and tasks are particular to the enterprise or job managed.

The universality of this management model allows us to apply it to project management of software development. Software development project management also involves the activities undertaken by one or more persons for purpose of planning and controlling the activities of others in order to achieve objectives that could not be achieved by the other acting alone. A project is a one-time effort having well-defined objectives and occurring within a specific time frame. A project plan is prepared, people are assigned to the project, resources are allocated, and success criteria are specified. A software development project is a project in which the objective is to produce a software product, on schedule and within budget, that satisfies a set of requirements[32].

Especially, project planning is the most important management activity. The project plan consists of two components; a development process which defines phases and the sequence of the phases, and the estimated period to execute each phase. The following sections discuss software development process and model for estimating development periods.

2.1 Software development process

2.1.1 Method and process

Figure 1 shows the relationships among a software development method, a software process, and the software project constraints. A method is a planned procedure by which a specified goal is approached step by step [15]. Most work

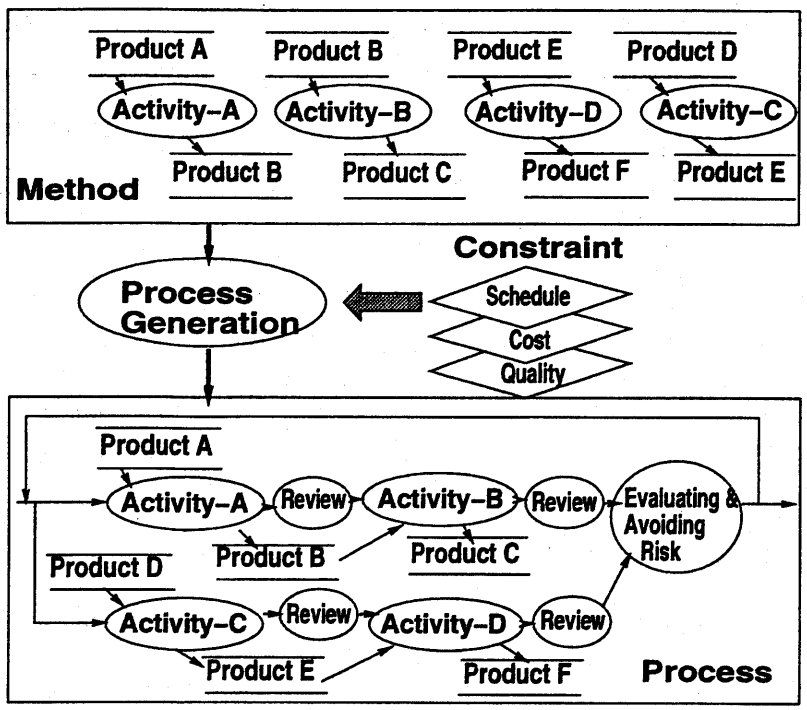

Figure 1. Relationship among a method, a process, and constraints

descriptions for program development are method descriptions. They describe, often in a very abstract manner, how one should think and reason in developing a software system, and products to be referred and/or created in these steps. Most methods also indicate the sequence of steps to be followed. A method is based on a preconceived notion of the architecture of the working system. This means that the description of the method is formulated in terms of the concepts of the architecture to be realized [15].

On the other hand, a process is generated from the method by taking constraints in real software development projects into consideration, e.g., constraints about cost, schedule, software quality and so on. For example; if we have to develop software in a relatively short period, we would generate a process including concurrent sub-processes among which activities are mutually independent and can execute concurrently. If we have to achieve high software reliability, we would generate an appropriate process by adding managerial activities, e.g., reviewing un-executable product and testing executable product to the method. If the user requirements are not clear and stable, we would generate an iterative process, e.g., a spiral process, in order to evaluate and avoid risk of requirement change.

The managerial aspect is one of the most important ones in the process generation, because we can consider the process to be a natural scaling-up of a method [15]. Let us consider an analogy. Producing a new chemical substance in the laboratory differs greatly from producing the same chemical on an industrial scale in a factory. In the laboratory, the goal is to find a method to produce the chemical. To make this method appropriate for large-scale industrial use, a process must be defined. This usually means changing the working method. Nobody would dream of industrializing the laboratory method by simply building a larger laboratory with gigantic test tubes and Bunsen burners [15].

2.1.2 Project milestones

Effective management relies on information. As software is intangible, this information can only be provided in the form of documents describing the work which has been carried out. Without this information, control of the project is lost, and cost estimations and schedules cannot be updated [30].

Progress control with milestones is one of the well-known approaches to managing software development. The date on which a work product is deemed to be completed is termed a milestone. In general, the milestones identified in a software development project correspond to points in time at which certain documents become available [34]:

- after requirements analysis, there is a requirement specification;
- after the design phase, there is a (technical) specification of the system;
- after implementation, there is a set of programs;
- after testing has been completed, there is a test report.

A typical example of a milestone is the date on which coding is completed and the codes pass reviews. Once a work product has been reviewed and agreed upon, it becomes a baseline product and can be changed only through formal procedures [28].

In order to determine whether a work product has reached a milestone, it must first pass a series of reviews performed by fellow team members, managers, and/or the clients [28]. Reviews at milestones should show whether the actual

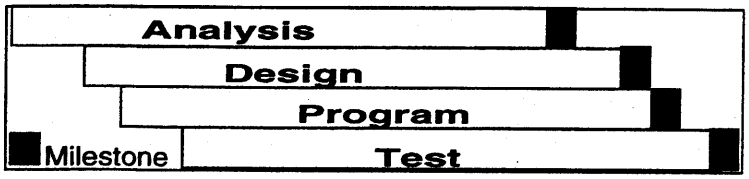

Figure 2. The object-oriented development process with traditional phases and milestones

work done is less than what was expected; whether there is a trend for milestones (representing completed work or reviews) to fail to be met; and whether the plans themselves keep changing so that completion dates are pushed further back or the time allocated to activities such as testing is being shortened to preserve the final delivery date [5].

2.1.3 Problems of control in object-oriented development projects

In the object-oriented approach, software development is inherently iterative and seamless [24]. That is, the same portion of the system is usually worked on a number of times with each iteration. Prototyping and feedback loops are standard. The seamlessness is accounted for in the lack of distinct boundaries between the traditional phases of analysis, design, and coding. The reason for removing the boundaries is that the concept of object permeates; objects and their relationships are the medium of expression for each of analysis, design and implementation [24].

The seamless nature of object-oriented development makes progress control with traditional milestones ineffective. If we apply milestones established for traditional software development to object-oriented projects then most such milestones are likely to be set in the second half of development (see Figure 2).

Several methods and tools have been proposed to generate development processes including project management activities. Katayama et al. proposed the Hierarchical and Functional Software Process (*HFSP*) model and applied it to the design process of a real project for developing a system for Artificial Spacecraft [16] [23]. They succeeded in extracting phases from the actual design process of the project, but they did not generalize the procedure of extracting phases and milestones from the project.

Hirayama et al. proposed a hierarchical model for software project which can evaluate the software process from the viewpoint of quality, cost and development period [12]. In this model, software development activity is expressed by an extended *Generalized Stochastic Petri-net* with some attributes. These attributes are used to represent the current status of software development, but a concrete procedure for progress control based on the current status of software development is not discussed.

Booch discussed a process model for object-oriented development process which provides a set of milestones for each phase [3]. However, he did not generalize the procedure of identifying phases, milestones, and baseline products.

Although we think that progress control with milestones is adaptable to object-oriented development, new milestones or new mechanisms of establishing milestones must be devised specifically for object-oriented development.

2.2 Resource estimation

2.2.1 Estimating methods

There are six primary methods used to develop software resources estimation[26].

- Analogy: Resource estimates are developed based upon past experience with similar systems.
- Bottom-up: Resource estimates are developed by tasks (design, code, test, etc.) or component level(program, release, delivery, etc) using analogy, PERT, parametric modeling or the Delphi technique.
- Parametric model: Resource estimates are developed using prediction models which mathematically relate effort and duration to those parameters which influence them.
- PERT: Resource estimates are developed assuming a normal or other (beta, etc.) probability distribution from estimates of the (l)worst possible, (m)most likely and (h)best possible effort or duration using the following formula; EFFORT=$(l + 4m + h)/6$

- Top-down: Resource estimates are allocated to specific WBS activities based on past experience.

- Delphi technique: Resource estimates are developed using a team of experts.

Depending upon the circumstances, each of these methods may have merit. On a small project, analogy method seems to work the best. For larger jobs, parametric models seem to do best because they codify experience and address scaling laws. For risky projects, Delphi is preferred because it allows you to use many experts to bound the risk via ranges[26].

2.2.2 Problems of estimating models

To estimate large projects, many parametric models and methods of modeling have been proposed. Although developers require much time to become familiar with a new development environment, these models and methods do not consider variation of productivity by developers' learning. Kellner proposed a method of modeling the software development process with three separate points of view: structural, functional, and behavioral[17]. The process simulation is based on the behavioral model mainly, and it executes STATEMATE which is one of the CASE tools used to design software for communication networks. It can establish the schedule for software development and can estimate work effort. Since STATEMATE is based on the behavioral model, productivity and learning curves might be calculated from probability of state-transition. But it does not take variations of productivity and learning curves into account. In addition, users need deep understanding to make the behavior model because the simulation can be executed after the users make the behavioral model.

Kusumoto et al. proposed a development model using an extended Generalized Stochastic Petri-net[19]. In this model, development period, work effort, and quality of software are estimated from assigning developers to the activities. The model has a parameter of the experience level of the developer. This parameter is important to determine the probability of the injection and removal of a fault and firing rate of a Petri-net. But the experience level parameter is constant for each developer, which does not change during the simulation. More, as the analysis needs good knowledge of the simulation model, the users require extensive

knowledge of the model to assign values to the parameters.

Iida et al. presented an overlapping development process model based on progress of activities[14]. In this model, development period and work effort can be estimated under overlapped activities. The model has a parameter which is the developer's ability. This parameter is significant to determine the productivity of the activity. But the developer's ability is also a constant for each developer and, it is not clear how values of the parameters should be determined.

3. Generation of Software Development Process

In this chapter, a new method for generating process is proposed as a new framework which gives us a guideline for generating ,especially, object-oriented development process with relevant milestones. Section 3.1 describes the framework. Application of the framework to two well-known object-oriented development methods is shown in section 3.2. Section 3.3 summarizes.

3.1 Proposed framework

We propose a framework which gives us a guideline for generating software development processes with relevant milestones for object-oriented development methods. As mentioned in section 2.1, the software development process should be generated from the development method and the specific project constraints. But we are not concerned here with the constraints in the process generation. It is because we believe that the main structure of the process can be constructed only based on the method and the constraints are used for customizing the generated process for the specific project.

The proposed framework consists of four steps: (1) *Method analysis*, (2) *Phase identification*, (3) *Baseline product identification*, and (4) *Process construction* (see Figure 3). The main point of this proposed framework is that software development phases and baseline products are identified based on the definition of software development method in an algorithmic way. That is, the proposed framework does not use traditional (and fixed) phases in generating software process, but establishes phases and baseline products customized to different types of development methods. The remainder of this section describes the four steps of the proposed framework.

(1) Method analysis

Method analysis is a procedure to clarify relationships among activities and products of the development method for which we generate software process. *Data Flow Diagram(DFD)* [8] is used to express the relationship. Activities and products of the method are expressed as "bubble" and "data stores" in *DFD*, respectively. An arc from a bubble to a data store means that the bubble produces

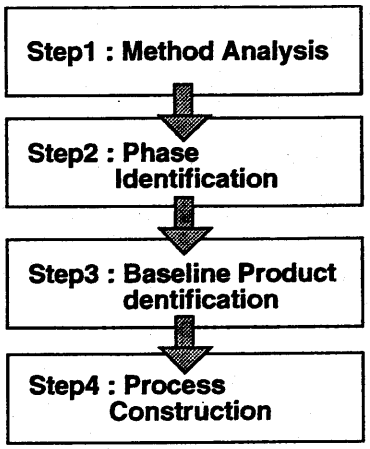

Figure 3. The proposed framework of generating software development process with milestones

the data store, i.e., the data store is the output product of the activity. On the other hand, an arc from a data store to a bubble means that the bubble refers to the data store, i.e., the data store is the input product of the activity. We believe that it is not so difficult to extract such information from the definition of the development method.

Figure 4 shows an example of *Method analysis* in which relationships among activities and products of the *Structured Development Method* [7] are depicted as a *DFD*. The *DFD* consists of eight activities and twelve products. The relationship among activities and products are relatively straightforward, which matches the fact that *Structured Development Method* is a traditional development method suitable to the waterfall process model.

(2) Phase identification

In the proposed framework, development phases are considered to be distinct stages of the project, at the end of which we can establish milestones. Some software development methods were developed with such phases already in mind. But most of the methods do not provide definite description of the phase. At first, the proposed framework clarifies the relationship between the activities and the phases.

To establish a milestone at the end of each phase, we do not allow any activities in a phase to create and/or update any products in the other phase. That is, an

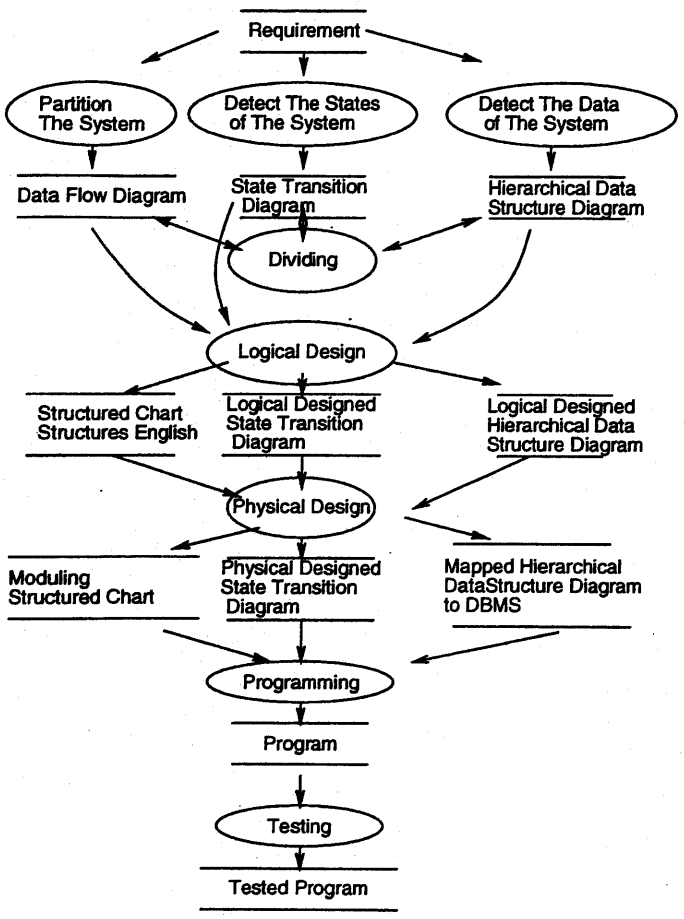

Figure 4. *DFD* of the *Structured Development Method*

activity can create or update products in the current phase but can only refer to products in the previous phases. Activities which create and/or update the same product must be members of the same phase. If this relationship between activities and products is satisfied, a formal progress report can be presented to management and a work product can be reviewed at the end of each phase in order to produce a baseline product which can only be changed through formal procedures in subsequent phases. In addition, the efforts of the reviews decrease since all we have to do is to review baseline products.

Figures 5 shows the input data structure, the output data structure, and the algorithm for phase identification. We call the phase identification algorithm *ALGORITHM1*. Input data *Dev_Method* corresponds to an array of the development activities defined in the development method and includes information about input and output products of each activity. We can easily get information for the input data from the *DFD* constructed in *Method analysis*. Output data *Dev_Process* corresponds to an array of the phase identified by *ALGORITHM1* and includes information about input and output products, activities of each phase, and order of phase execution. In *ALGORITHM1*, first, development activities with common output products are collected into a set. Each set corresponds to a phase to be identified. Then the order of phase execution is determined.

Figures 6 and 7 show the result of the application of *ALGORITHM1* to *Structured Development Method*. Five phases were identified: Analysis, Logical Design, Physical Design, Programming, and Testing. From the result, we can conclude that the waterfall process model is suitable to *Structured Development Method*, since these phases have to be executed sequentially.

(3) Baseline product identification

We only have to check whether a product is referred to by activities in the subsequent phase in order to identify the baseline products among all products in the phase identified by *ALGORITHM1*. It is because that the baseline products of a phase can be changed in the subsequent phases only through formal procedures [28], and *ALGORITHM1* can identify phases in which no product is updated by activities in subsequent phases.

Figure 8 shows the algorithm for baseline product identification along with its input data structure and its output data structure. We call the baseline product

```
Input:(* Software development method information *)
     Dev_Method:array[1..act_max] record of
            {   Input_Products:set;
                Output_Products:set;
                Activity:string;   }
Output:(* The generated process information in algorithm 1*)
     Dev_Process:array[1..phase_max] record of
              {  Input_Products:set;
                 Output_Products:set;
                 Activities:set;
                 Next_phases:set;}
algorithm1:
  index:=0;
  for i:=1 to act_max do
  { if Dev_Method[i] ≠ nil{
      index:= index+1;
      Add members of Dev_Method[i] to members of Dev_Process[index].

      for j:=i+1 to act_max do
      { if (Dev_Method[i].Output_Products ∩
                  Dev_Method[j].Output_Products) ≠ φ {
          Add members of Dev_Method[j] to members of Dev_Process[index].
          Dev_Method[j]=:nil;}
      }}
  }
  for i:=1 to phase_max do
  { for j:=i+1 to phase_max do
    { if (Dev_Process[i].Output_Priducts ∩
                  Dev_Process[j].Input_Products) ≠ φ
            Add Dev_Process[j] to Dev_Process[i].Next_phases.
    }
  }
```

Figure 5. *ALGORITHM 1*: Algorithm for phase identification along with its input and output data structures

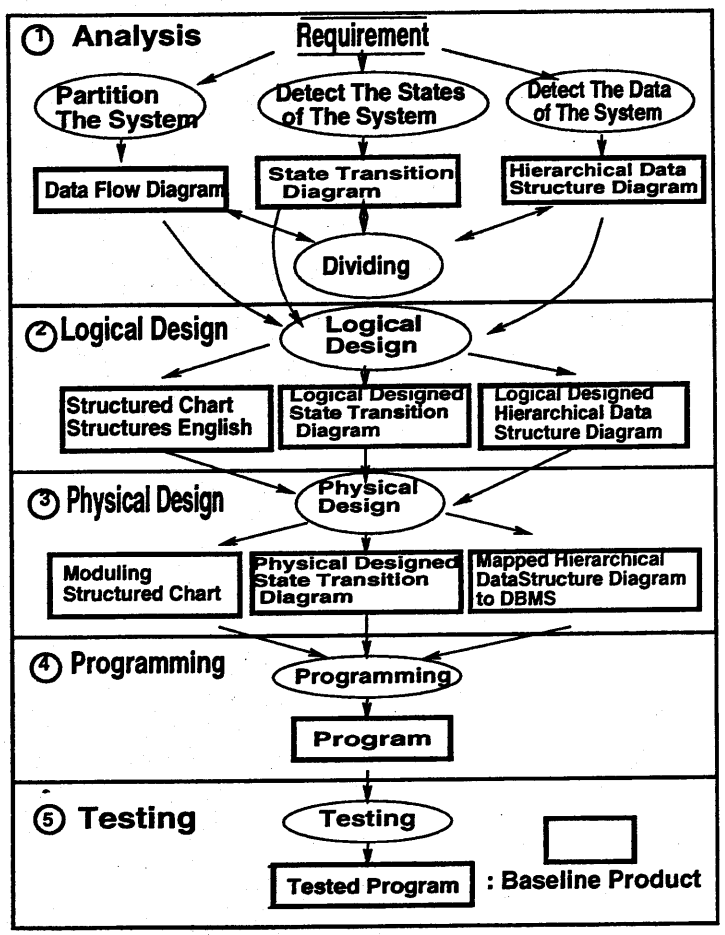

Figure 6. Phases of the *Structured Development Method* identified by *ALGORITHM 1*

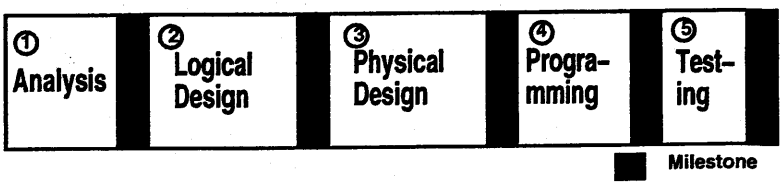

Figure 7. The Phase Sequence of the *Structured Development Method* identified by *ALGORITHM1*

```
Input:(* The generated process information in Algorithm1 *)
  Dev_Process:array[1..phase_max]

Output:(* The milestone information *)
       Dev_milestone:array[1..milestone_max] record of
                     {
                        milestone_date:DATE;
                        Baseline_Products:set;
                        check_phase:index;
                     }

algorithm2:
  for i:=1 to phase_max
  {
    Dev_milestone[i].check_phase:=i;
    for j:i+1 to phase_max
    {
      if (Dev_Process[i].Output_Products ∩
                 Dev_Process[j].Input_Products) ≠ φ then{
          Add (Dev_Process[i].Output_Products ∩
                        Dev_Process[j].Input_Products)
              to Dev_milestone[i].Baseline_Products.}
    }
  }
```

Figure 8. *ALGORITHM 2*: Algorithm for baseline product identification along with its input and output data structures

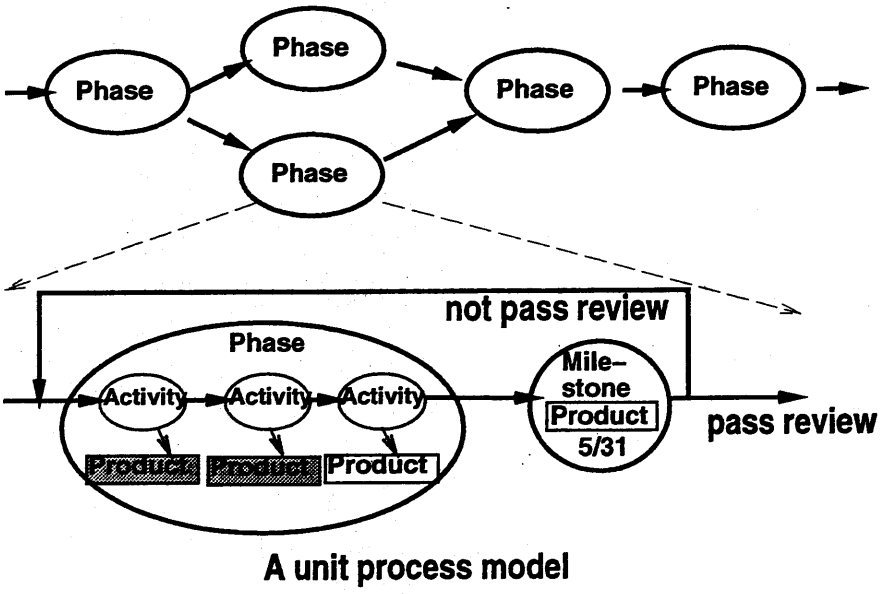

Figure 9. A unit process model of the *phase and milestone based management process*

identification algorithm *ALGORITHM2*. Input data *Dev_Process* is the same as the output data of *ALGORITHM1*. That is, *ALGORITHM2* receives the output data of *ALGORITHM1*. Output data *Dev_milestone* corresponds to an array of the milestone of each phase and includes information about the baseline product identified by *ALGORITHM2* and the milestone date which will be set during the project planning. *ALGORITHM2* compares input and output products of all activities, and finds the products to which are referred by the subsequent phases.

For the *Structured Development Method* shown in Figure 6, the application of *ALGORITHM2* results in all products being baseline products except for the initial work product "Requirement" which is an input to the software process. But the initial products must be checked by hand before starting the software process, and may be changed only through formal procedures. Thus, it can be considered as a baseline product. Consequently, all products of the *Structured Development Method* are baseline products. This is another good evidence to prove that the waterfall process model is suitable to *Structured Development Method*.

(4) Process construction

Process construction is a procedure that builds a concrete software process for

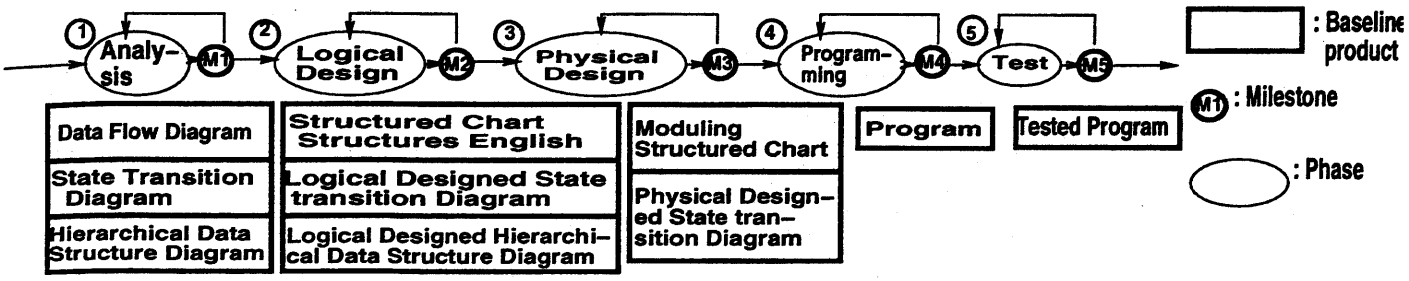

Figure 10. A model of the *phase and milestone based management process* for the *Structured Development Method*

the corresponding method including managerial activities to control development progress based on phases and milestones. We call such a software process *phase and milestone based management process.* A unit process model of the *phase and milestone based management process* consists of a phase, a milestone and baseline products as shown in Figure 9. In the current version of the unit process model, we have to re-enter the phase if we cannot pass review done at the milestone for the phase.

We have to replace an identified phase with a unit process model. Then we can build a concrete *phase and milestone based management process* as shown in Figure 10. It describes the order of phase execution and shows the products that must be checked at each milestone.

To implement and execute the *phase and milestone based management process* in real software projects, we need a more detailed description about the process. In the proposed framework, a detailed description of the *phase and milestone based management process* are given in the form of *OMT* documents. Figures 11, 12, 13 and 14 are the object diagram, event trace diagram, state diagram, and data flow diagram of the *phase and milestone based management process*, respectively.

3.2 Application

In this section, we present examples of the application of the proposed framework. We tried to generate software processes for two well-known object-oriented devel-

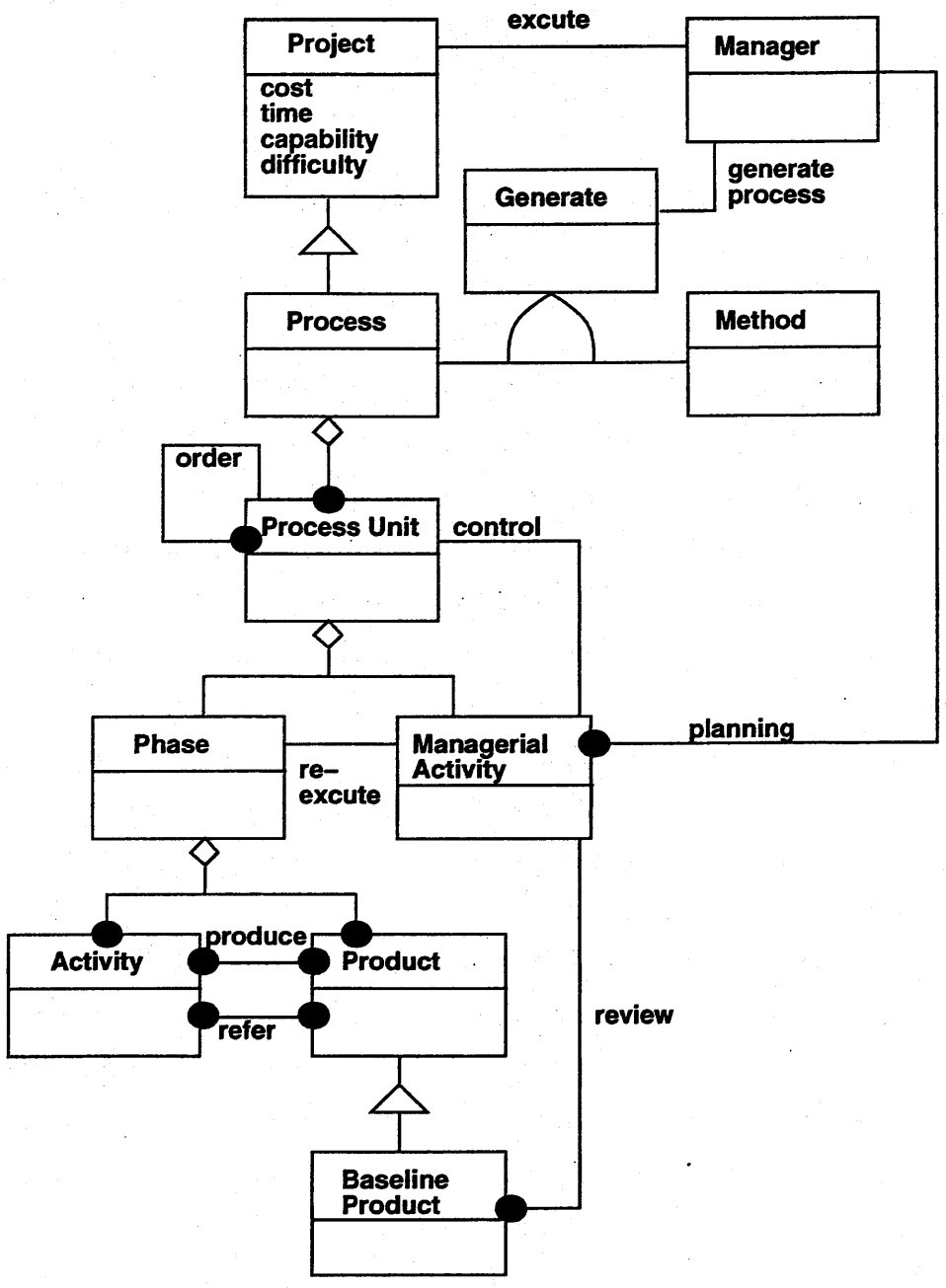

Figure 11. The object diagram of *phase and milestone based management process*

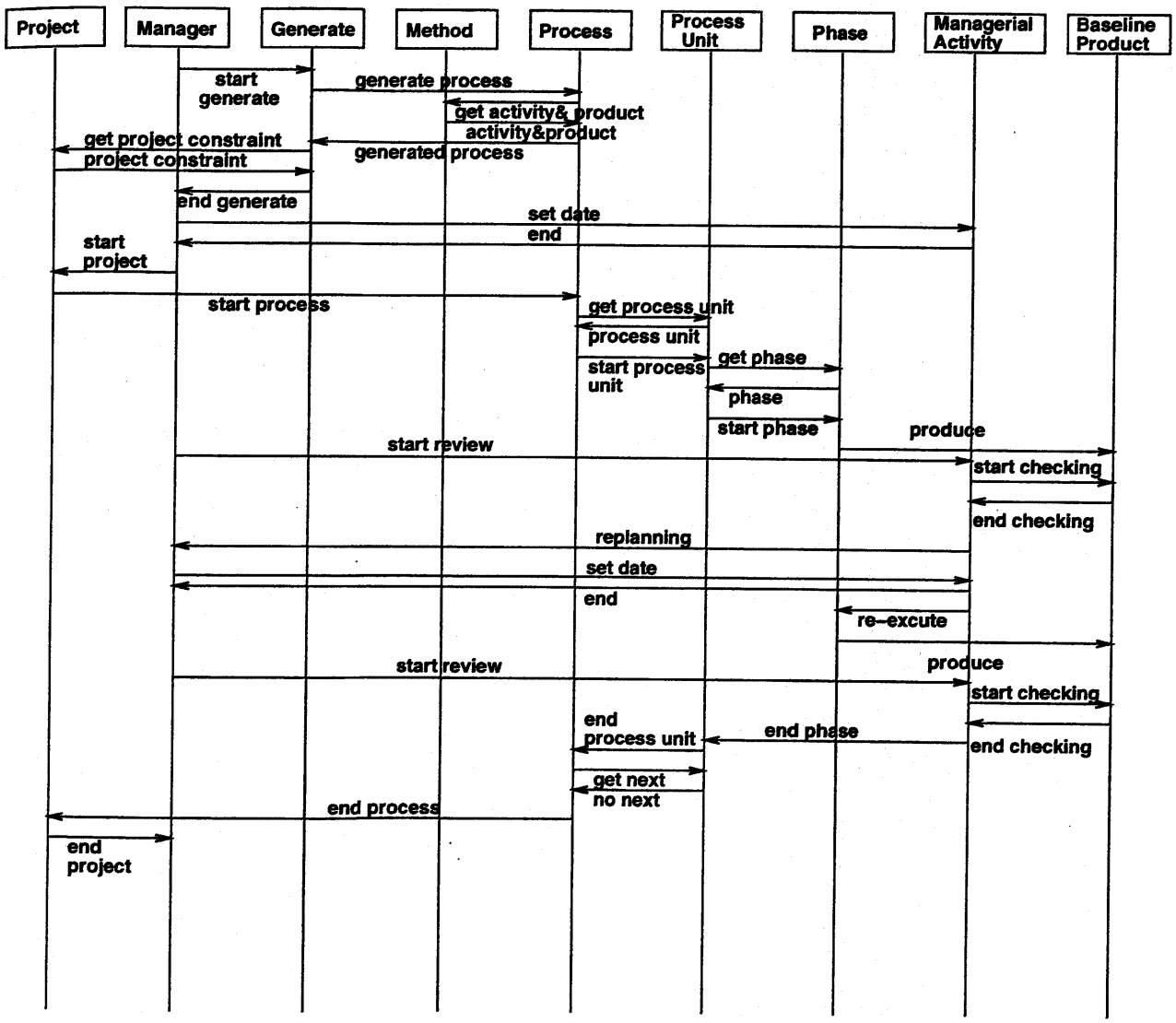

Figure 12. The event trace diagram of *the phase and milestone based management process*

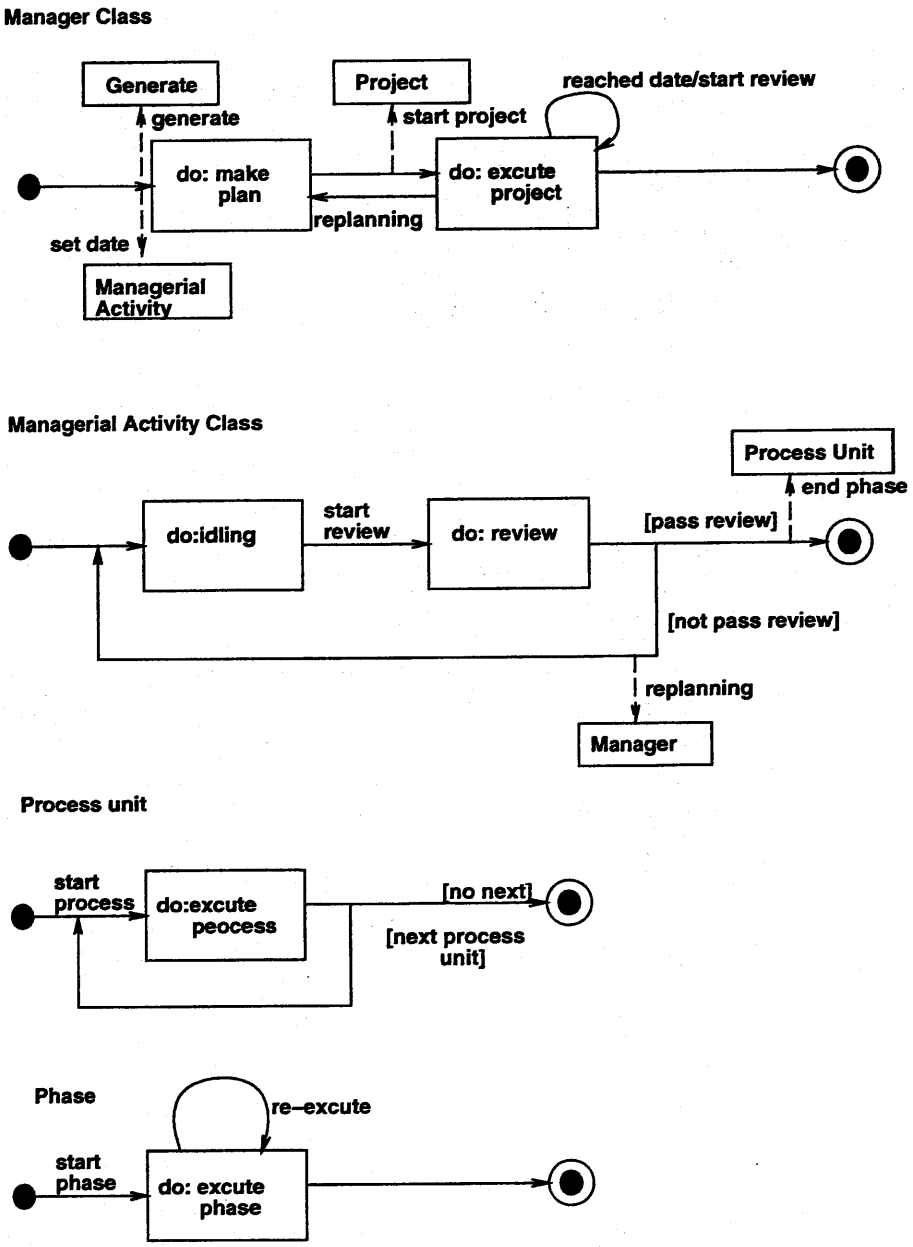

Figure 13. The state diagram of *the phase and milestone based management process*

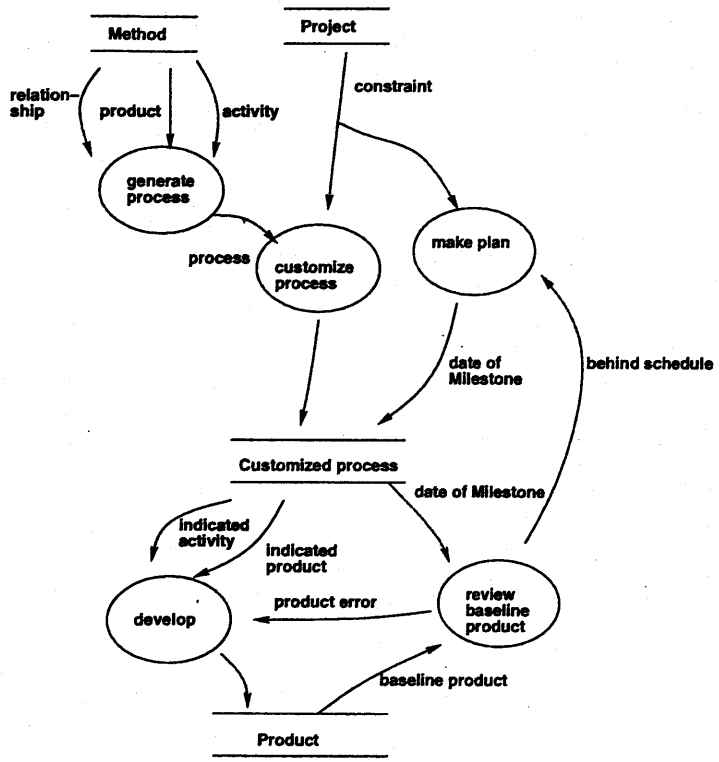

Figure 14. The data flow diagram of *the phase and milestone based management process*

opment methods: *Object Modeling Technique* (*OMT*) development method and *Object Oriented Software Engineering* (*OOSE*) development method. In these examples, it is assumed that we cannot divide any products used in the development method into smaller units such as modules and files.

(1) Example 1 : *OMT*

Figure 15 shows the *DFD* of *OMT*. The phases and the baseline products identified by *ALGORITHM1* and *ALGORITHM2* are also shown in Figure 15. Figure 16 shows the phase sequence of *OMT* identified by *ALGORITHM1*. In addition, Figure 17 depicts the general model of the *phase and milestone based management process* customized for *OMT* by the proposed framework. It is not so difficult to construct the *DFD* shown in Figure 15 by hand based on the description of *OMT* [27].

As shown in Figure 15, fifteen activities are classified into six phases and seven baseline products are identified out of sixteen products. The number of activities varies among the phases: phase #1 and phase #5 consist of five and six activities, respectively, and each of the other four phases consists of only one activity. The number of products also varies among the phases: phase #1 and phase #5 both include six products, respectively, and each of the other four phases includes only one product. If there is an activity which refers to and updates many products referred by other activities, the number of activities and/or products of the phase tends to be large. For example, in phase #1, "Deciding behavior of system" activity refers to and updates four products which are the output products of the other four activities of the phase (see Figure 15). Such variations in the number of activities and/or products among the phases results in variety of phase lengths.

If we intend to shorten the length of a phase, we would need to introduce some restrictions on the updating of products by activities of the phase. For example, if we prohibit the "Deciding behavior of system" activity from updating "Object class of system", "Object diagram" and "Object diagram with increased inheritances", we can divide phase #1 into four distinct phases and consequently nine phases are identified for the whole process as shown in Figures 18 and 19.

(2) Example 2 : *OOSE*

Figure 20 shows the *DFD* of *OOSE*. The phases and the baseline products identified by *ALGORITHM1* and *ALGORITHM2* are also shown in Figure 20. Figure

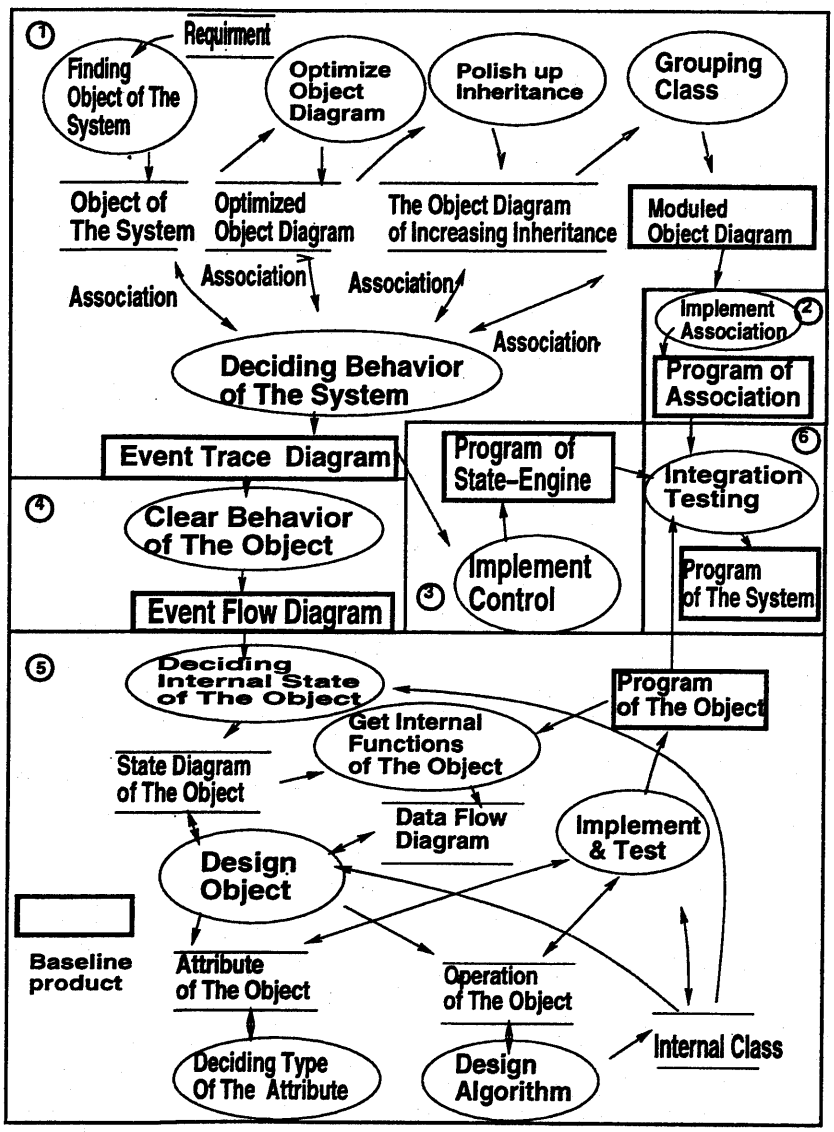

Figure 15. Phases and baseline products of *OMT* identified by *ALGORITHM1* and *ALGORITHM2*

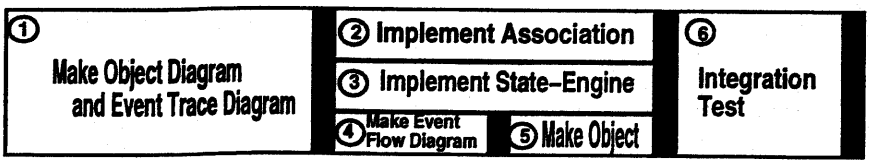

Figure 16. The phase sequence of *OMT* identified by *ALGORITHM 1*

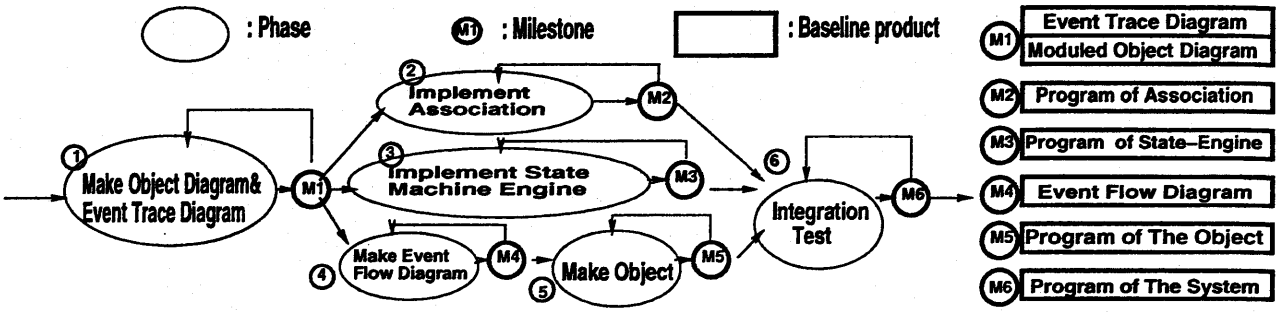

Figure 17. A general model of *the phase and milestone based management process* for *OMT*

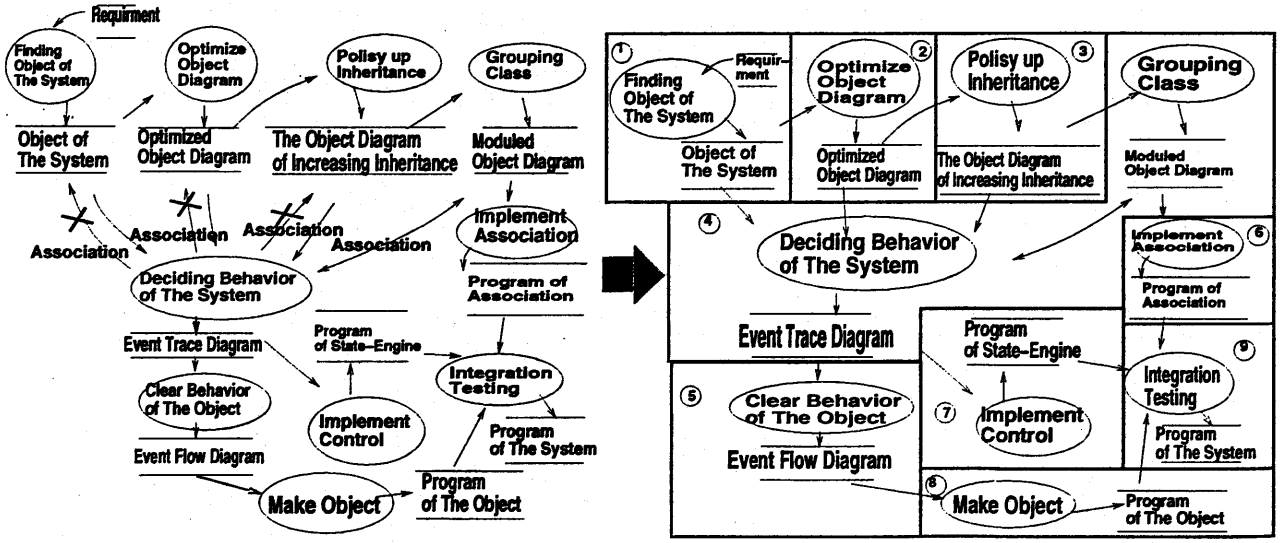

Figure 18. Phase of the modified *OMT*

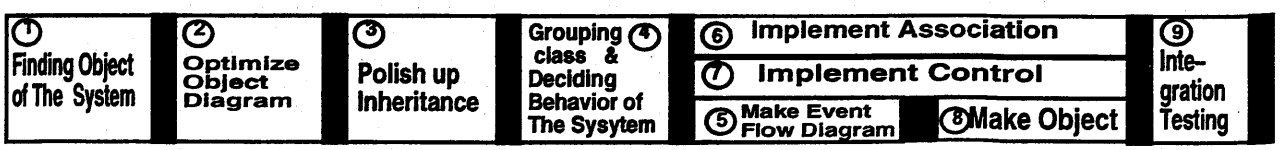

Figure 19. The phase sequence of the modified *OMT*

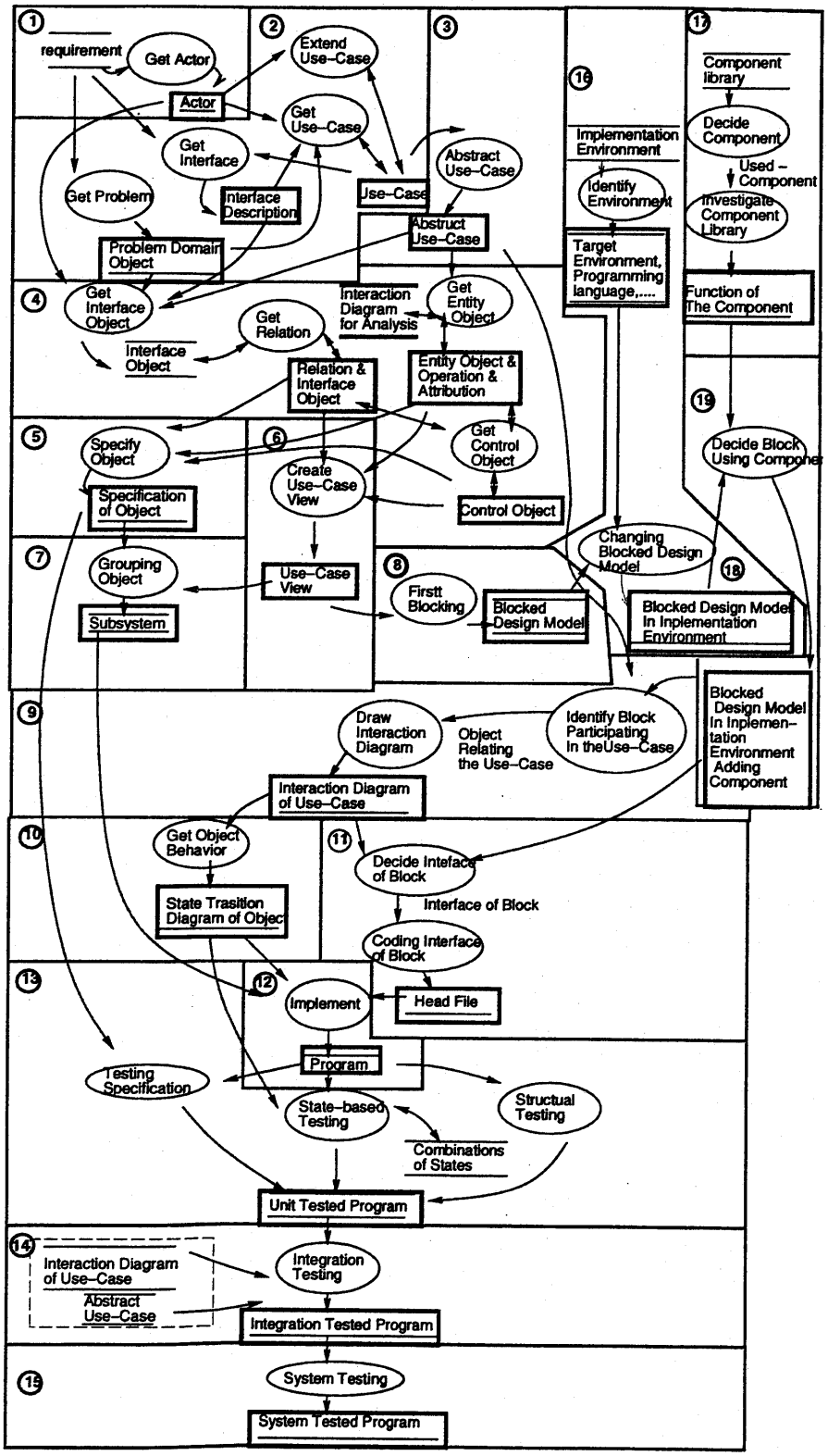

Figure 20. Phases and baseline products of *OOSE* identified by *ALGORITHM1,2*

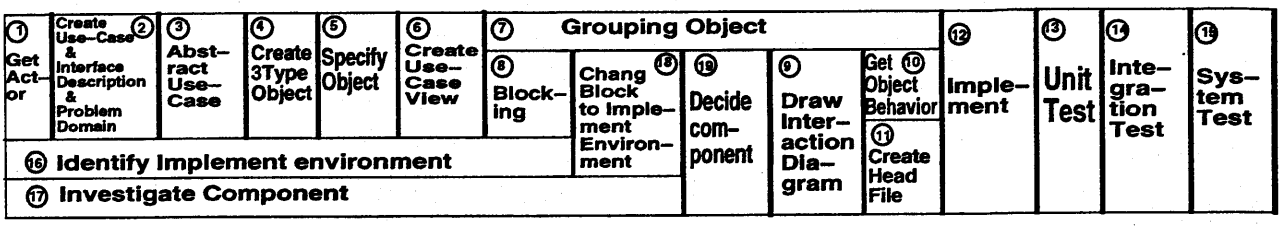

Figure 21. The phase sequence of *OOSE* identified by *ALGORITHM1*

21 presents the phase sequence of *OOSE* identified by *ALGORITHM1*. In addition, Figure 22 depicts the general model of the *phase and milestone based management process* customized for *OOSE* by the proposed framework. It is not so difficult to construct the *DFD* shown in Figure 20 by hand based on the description of *OOSE* [15].

As shown in Figure 20, thirty activities are classified into nineteen phases and twenty-three baseline products are identified out of thirty-one products. The number of phases and baseline products are relatively large compared with *OMT*. Although a large number of phases and/or baseline products results in precise control of development progress, it also results in additional managerial cost and strongly rigid development process. If we intend to reduce the number of phases and/or baseline products, we need to integrate successive phases. There are three criteria for determining which pair of phases is appropriate for being integrated:
(C1) Length of phase $\cdots$ Milestones established at irregular intervals prevent effective control of development progress. We have to avoid increasing the variation in the length of the phases by phase integration.
(C2) Reliability of baseline product $\cdots$ When we integrate two successive phases into one, the baseline product of the former phase is referred by the activity in the latter phase without checking its correctness. We have to avoid increasing the probability of rework by phase integration.
(C3) Concurrency of phase execution $\cdots$ When we integrate two successive phases into one, the baseline product of the former phase is not authorized as a "real" baseline product for other phases to refer until the baseline product of the latter phase has been authorized. Therefore, we can not concurrently execute an integrated phase and phases which refer to the baseline product of the former

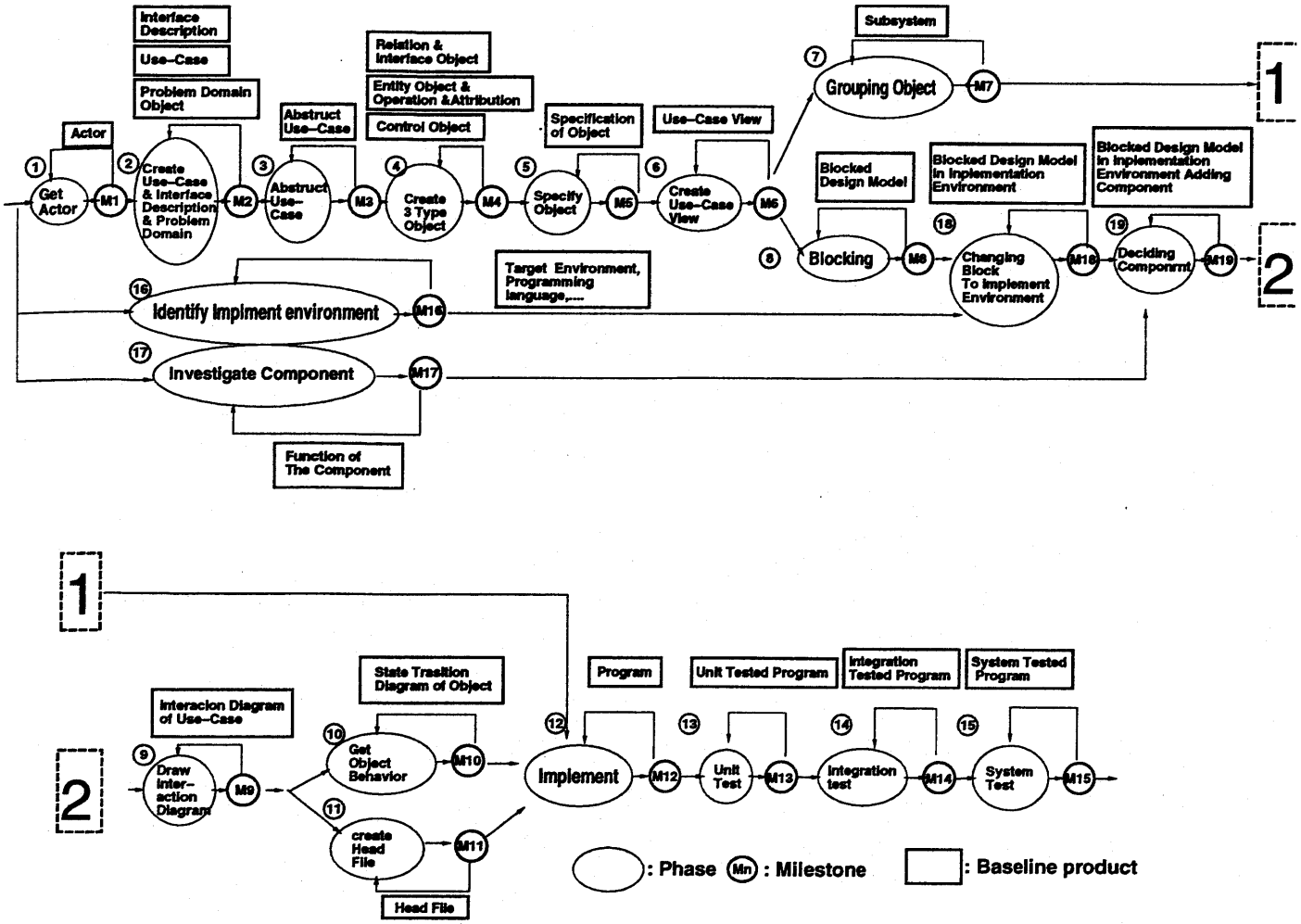

Figure 22. A general model of the phase and milestone based management process for *OOSE*

phase. For example, if we integrate phase #6 and phase #8 in Figure 21, we could not start phase #7 until the integrated phase of phase #6 and phase #8 completes. We have to avoid decreasing the concurrency of phase execution by phase integration.

Such adjustment of length and/or the number of phases discussed in this section may be reasonable from the viewpoint of project management. But there is the possibility that the adjustment may be against the principle of the development method and may not be acceptable to all managers and/or developers of the project.

3.3 Conclusion

This chapter proposed a new framework which gives us a guideline for generating software processes with relevant milestones for object-oriented development methods. In the proposed framework, phases and baseline products can be identified in an algorithmic way based on the definition of the development method. This chapter also presents examples of the application of the proposed framework, in which we generated software processes for two well-known object-oriented development methods: *OMT* and *OOSE*. The examples show that the framework can generate a software process customized for object-oriented development methods in a systematic way. And discussion for adjustment of the length and the number of phases is shown.

The framework can be applied not only to object-oriented development methods but all development methods where the relationships among activities and products can be expressed in a *DFD*. In addition, the proposed framework can be applied to the incremental development approach and the spiral development approach. It is because each iteration process of these approaches can be controlled by phases and milestones. We believe that the framework has potential capability to generate software process with relevant milestones for various kinds of software development methods and approaches.

4. Estimating Model based on developer's learning

In this chapter, a new simulation model is proposed. The model is useful for estimating development periods in project planning. Especially, the model can estimate and/or predict development periods which are influenced by variation of developers' productivity. The productivity varies with developers' learning. Section 4.1 describes the proposed model, applications of the model is shown in section 4.2. Customization and evaluation of the model are described in section 4.3. Section 4.4 presents a new simulator based on the proposed model. Case studies on the simulator are also shown. Section 4.5 summarizes.

4.1 The proposed model

4.1.1 The three submodels

The proposed model consists of three submodels: the Activity model, the Productivity model, and the Knowledge model. The relationship of the three submodels is shown in Figure23. The Activity model presents characteristics of an activity. The Knowledge model presents characteristics of a developer. The Productivity model presents the relationship between the activity and the developer, by using the developer's productivity during the execution of the activity.

(1)The Activity Model

In the proposed model, an activity of development is composed of a set of simple activities called "primitive activity." For example, a design activity is composed of a set of primitive activities: architectural design, interface design, component design, data structure design, and algorithm design[30]. A primitive activity has a knowledge level which is needed to execute this primitive activity. The model is based on the following assumptions:

(i) an activity is set up on each project,

(ii) an activity is composed of a set of primitive activities,

(iii) the primitive activities are put in order of the level of the required knowledge,

(iv) the high knowledge level requires more knowledge than the low knowledge level, and

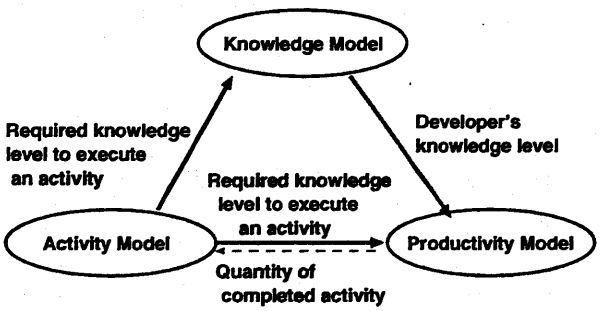

Figure 23. Outline of the proposed model

(v) the knowledge needed to execute a primitive activity of higher rank includes all knowledge needed to perform a primitive activity of lower rank.

The Activity model shows the relationship between the primitive activity's quantity and the required knowledge level needed to execute the primitive activity. The Activity model presents a distribution of knowledge levels needed to execute the primitive activities(See Figure 24). The X axis of the Activity model in Figure 24 means the knowledge levels needed to execute the primitive activities. In Figure 24, the knowledge levels of the Activity model are divided into 256 levels. The Y axis in Figure 24 means the total amount of the primitive activities which require the same knowledge level. Assuming that the distribution of knowledge level is normal distribution, the Activity model is defined as follows:

$$W_j(\theta) = w_j \frac{1}{\sqrt{2\pi}\sigma} e^{\left[-\frac{(\theta-\mu)^2}{2\sigma^2}\right]} \tag{1}$$

where

$W_j(\theta)$: total amount of primitive activities which require knowledge level θ under an activity j

θ : required knowledge level to execute a primitive activity of activity j

w_j : total amount of activity j

μ : average of θ

σ : standard deviation of θ

(2) The Productivity model

The Productivity model shows a developer's productivity in the execution of an activity. When the developer does a primitive activity A, the developer's productivity is derived by processing the developer's knowledge level and the required

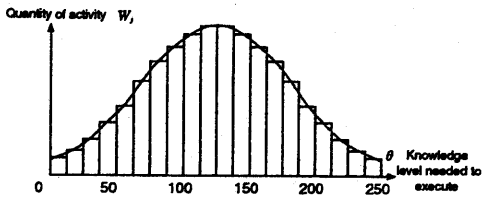

Figure 24. Activity model

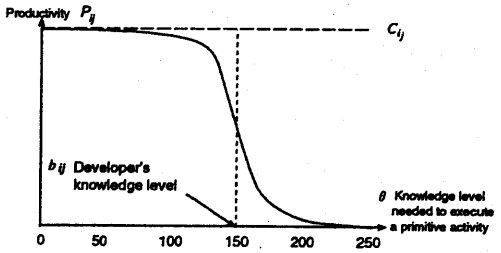

Figure 25. Productivity model

knowledge level needed to execute the primitive activity A. The Productivity model using a cumulative normal model *Ogive model*[20] is defined as follows(See Figure 25):

$$P_{ij}(\theta) = C_{ij} \int_{-\infty}^{a_j(b_{ij}-\theta)} \frac{1}{\sqrt{2\pi}} e^{\frac{-t^2}{2}} dt \tag{2}$$

where

$P_{ij}(\theta)$: productivity of a developer i under a primitive activity of an activity j, which has knowledge level θ

C_{ij} : maximum of the productivity of developer i under activity j

a_j : level of accuracy needed to execute activity j (≥ 0)

b_{ij} : knowledge level of developer i about activity j

θ : required knowledge level to execute the primitive activity of activity j

In Figure 25, the X axis of the Productivity model means the knowledge levels needed to execute the primitive activities. The knowledge levels of the Productivity model are also divided into 256 levels. The Y axis means a developer's

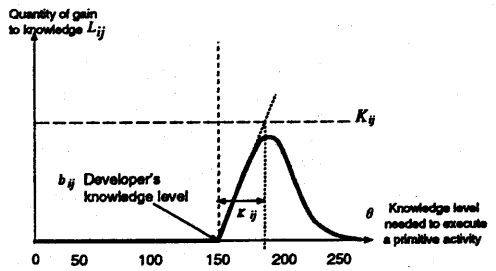

Figure 26. Knowledge model

productivity. The variation of the productivity is shown as a curve in Figure 25. Figure 25 depicts that, if the developer's knowledge level b_{ij} is higher than the required knowledge level θ, the productivity is high. In contrast, if b_{ij} is less than θ, the productivity is low. Productivity, especially, decreases in narrow limits. The range is around a point where θ is almost equal to b_{ij}. And if θ is equal to b_{ij}, the productivity is a half of the productivity's maximum C_{ij}.

The parameter a_j is very important for determining the shape of the curved line in the model[20]. If a_j is big, the curved line declines sharply within narrow limits, a small gap in the developer's knowledge can greatly change the productivity. On the other hand, if a_j is small, the curved line goes down loosely within wide limits. In particular, if a_j is equal to 0, the productivity is always a half of the maximum productivity C_{ij} regardless of value of θ; a large gap in the developer's knowledge will only change the productivity slightly. For example, if a developer does not know a "ls" command on text-command based OS such as UNIX, he/she will not be able to display a list of files. In the UNIX OS, the value of the parameter a_j is big because the developer is required accurate knowledge about UNIX commands such as "ls". In contrast, even if a developer does not know how to display a list of files on GUI based OS such as Windows, he/she will be able to display the list in some trials of clicking menus and buttons. In the Windows OS, the value of the parameter a_j is small because the developer does not have to have accurate knowledge about the operational functions of listing files.

(3) The Knowledge model

The Knowledge model shows quantity of gain to a developer's knowledge by executing an activity. Quantity of gain to the developer's knowledge is derived

from the relationship between b_{ij} and θ. b_{ij} is the developer's knowledge level and θ is the required knowledge level to execute the activity. This model is based on the following assumptions,

- if b_{ij} is more than θ, developer i will not gain new knowledge by executing the primitive activity, the developer's knowledge level is not changed.
- if b_{ij} is less than θ, developer i will gain new knowledge by executing the primitive activity, the developer's knowledge level is increased. Provided that, if the gap between b_{ij} and θ is small, developer i gains more new knowledge; if the gap is large, developer i gains limited new knowledge. In short, the developer has to do a difficult activity for him/her to gain knowledge. But if the activity is too difficult for him/her, he/she gains little knowledge.

The Knowledge model is defined as follows(See Figure 26):

$$L_{ij}(\theta) = \begin{cases} (\theta - b_{ij}) K_{ij} e^{-E_{ij}(\theta - b_{ij})} & (b_{ij} \leq \theta) \\ 0 & (b_{ij} > \theta) \end{cases} \tag{3}$$

where

$L_{ij}(\theta)$: quantity of gain to knowledge of a developer i by executing a primitive activity of activity j, which has knowledge level θ

K_{ij} : maximum quantity of gain to knowledge of developer i by executing activity j

b_{ij} : developer i's knowledge level about activity j

E_{ij} : developer i's efficiency of gain to knowledge by executing activity j

θ : required knowledge level to execute the primitive activity of activity j

4.1.2 Simulation using the proposed model

The proposed model can be used to simulate a development progress. The following equation has been applied to compute the value of progress.

$$Progress = \frac{w_j - \Sigma_\theta W_j(\theta)}{w_j} \tag{4}$$

where

$W_j(\theta)$: remaining quantity of primitive activities which require knowledge level θ under an activity j

θ : the required knowledge level to execute primitive activities of an activity j

w_j : total amount of activity j'

This simulation consists of the following steps:

Step 0 Initialization

Parameters(w_j,b_{ij},μ,σ,a_j,b_{ij},C_{ij},K_{ij},E_{ij}) of the three submodels are initialized to the values determined by a user(e.g., a project manager). The time t is initialized to zero.

Step 1 Choice of a primitive activity

A primitive activity which is executed at time t is chosen randomly out of the primitive activities with non-zero quantities in the Activity model. Parameter θ is determined by the chosen primitive activity.

Step 2 Calculation of productivity P_{ij}

The value of Productivity P_{ij} is calculated by Equation(2) in the Productivity model. The Productivity model is given the values of the four parameters: C_{ij} which is determined on Step0, a_j which is determined on Step0, b_{ij}(developer's knowledge level) which is determined on Step0(only in the first cycle) or Step5, and θ(required knowledge level to execute the primitive activity) which is determined on Step1.

Step 3 Renewal of the quantity of a primitive activity $W_j(\theta)$ in the Activity model

P_{ij} which has already been calculated on Step2 is subtracted from the amount of the primitive activity which has the θ (required knowledge level to execute the primitive activity). The Activity model is reset to the result of this subtraction. Using mathematical term, it could be expressed as:

$$W_{j(\theta)}(t+1) = W_{j(\theta)}(t) - P_{ij(\theta)} \tag{5}$$

Step 4 Calculation of quantity of gain to knowledge L_{ij}

Quantity of gain to knowledge L_{ij} is calculated by Equation(3)in the Knowledge model which is given four parameters: K_{ij} which is determined on

Step0, E_{ij} which is determined on Step0, b_{ij}(developer's knowledge level) which is determined on Step0(only in first cycle) or Step5, and θ(required knowledge level to execute the primitive activity) which is determined on Step1.

Step 5 Renewal of the developer's knowledge level b_{ij}, and quantity of gain to developer's knowledge in the Knowledge model
The quantity of gain to knowledge L_{ij} which has already been calculated on Step4 is added to b_{ij} (the developer's knowledge level).

$$b_{ij}(t+1) = b_{ij}(t) + L_{ij}(t) \tag{6}$$

The knowledge level is reset to the developer's new knowledge level b_{ij}.

Step 6 Renewal of value of productivity in the Productivity model
The value of productivity is reset to the developer's new knowledge level b_{ij} determined on Step5.

Step 7 Goto Step1
In the Activity model, if the total amount of remaining activities $\Sigma_\theta W_j$ is equal to 0, the simulation is finished; if $\Sigma_\theta W_j$ is greater than 0, set the value of time t to $(t+1)$, then go back to Step1.

In the simulation, the developer's knowledge level b_{ij} is increased at Step 5. The growth of the developer's knowledge level b_{ij} during the execution of activity j shows the developer's learning curve. Line(1) in Figure 27 shows the learning curve in the simulation. The growth of the developer's knowledge level b_{ij} has a great impact on the development progress.

The shape of the learning curve depends on four factors: E_{ij} and K_{ij} in the Knowledge model, the way that the primitive activity is chosen (e.g., at random on Step1), and the shape of the Activity model(e.g., normal distribution). For example, if the primitive activity is chosen in ascending order of the required knowledge level, the shape of the learning curve in early stage of the simulation will be flat (See Line(2) in Figure 27). This is because that the L_{ij} which is calculated on Step 4 is equal to 0, when the b_{ij} is greater than the θ of the chosen primitive activity. In the latter half of the simulation, the shape of the learning

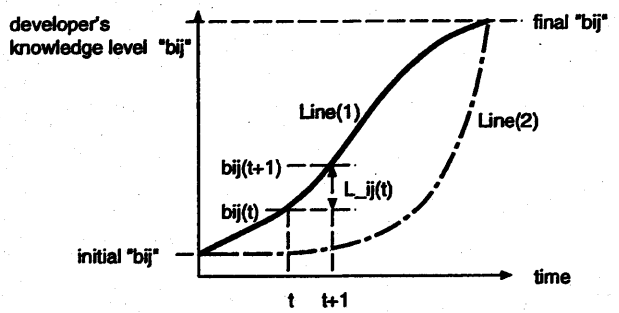

Figure 27. Learning curve in the simulation

Table 1. The default values of the parameters

total amount of activity $j(W_j)$	5000
developer i's knowledge level(b_{ij})	125
average of the required knowledge level to execute primitive activities:$\theta(\mu)$	125
maximum of the developer i's productivity (C_{ij})	1
distribution of the required knowledge level to execute primitive activities:$\theta(\sigma)$	40
developer i's maximum quantity of gain to knowledge (K_{ij})	1
level of accuracy needed to execute the primitive activities(a_j)	1
developer i's efficiency of gain to knowledge by executing(E_{ij})	0.02

curve will go up sharply. This is because that the L_{ij} which is calculated on Step 4 is almost equal to K_{ij} which is maximum gain to knowledge, when b_{ij} is slightly less than the θ.

4.2 Case study

In order to make clear the usefulness and characteristics of the proposed model, we applied this model to four cases in which the development progress is influenced by the characteristics of the activities and the developers. Table 1 shows the default values of the parameters. In the following applications, we would refer to these values unless further notices are given. The value of b_{ij} is between 0 and maximum 250. The default knowledge level is set to 125, the

sufficient knowledge level is set to 175, and the insufficient knowledge level is set to 75.

Case1: Three developers with different knowledge levels perform the same activity.

In this case, we assume that the developers design software using a design method which only used simple notations. The developers can design the software even if they understand only the outline of the method and the notations. But it takes a lot of training to design efficiently, which means that the design satisfies user's requirements, and has few defects.

The first scenario is that a developer starts to design after studying the method sufficiently. He/she can design with no defects. The second scenario is that a developer starts to design after studying the method on average. He/she can design with some defects. The third scenario is that a developer starts to design with a little knowledge about the method. He/she can design with many defects. All developers learn the method while they design the software using the method.

The developers' progress is expected to be different in the three scenarios. The proposed model has been evaluated against its ability to indicate the progresses gap between three developers. Three scenarios are as follows:
(1-1) with sufficient knowledge($b_{ij} = 175$),
(1-2) with average knowledge($b_{ij} = 125$),
(1-3) with insufficient knowledge($b_{ij} = 75$).

Figure 28 shows the result of these scenarios. In this result, the developer with sufficient knowledge makes greater progress in the activity, the developer with insufficient knowledge makes poor progress. This outcome is in line with software development intuition. If some information about the effort of the training is given, the developer's training period for the method before he/she starts to design can be obtained.

By the same token, the proposed model shows a sudden change of productivity's incline at around 90% in scenario(1-1)(Figure 28). There is a great variation of the productivity of the experienced developer at different time blocks, one is in the beginning, one is at the end of the development period. This gap indicates the decrement in the developer's productivity. For scenario(1-1), it is not necessary to learn while the developer designs the software. However, as a few primitive

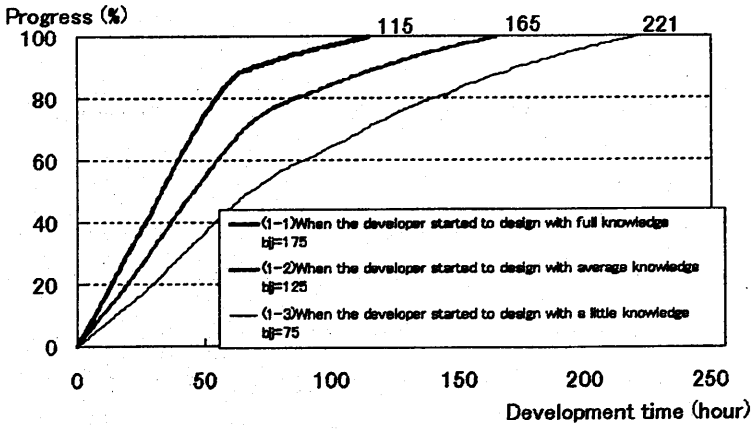

Figure 28. Result of Case1

activities which have greater θ than his/her b_{ij} have been left over at the end of the development period, the developer's productivity P_{ij} is small. At the end of the development, it should be time consuming for the developer to learn when doing such difficult primitive activities.

Case2: Two developers with different knowledge levels and different learning curves execute the same activity.

The learning curve of a new programming language depends on the individual's knowledge level of the language theories and the experience with using other similar programming languages. If there is a little knowledge about the language theories and experience, it would be difficult for the developer to acquire more knowledge.

We evaluated whether the proposed model had an ability to indicate a progress gap between two developers. One developer has a high knowledge level of programming language, but he/she can hardly learn the new technology or environment. Another developer has a low knowledge level of programming language, but he/she can learn the new technology or environment easily.

(2-1) The developer has a high knowledge level of programming language, but he/she has difficulties with learning new skills. ($b_{ij} = 150, K_{ij} = 1, E_{ij} = 0.01$)

(2-2) A developer has a low knowledge level of programming language, but he/she has little difficulties with learning new skills. ($b_{ij} = 100, K_{ij} = 10, E_{ij} = 0.1$)

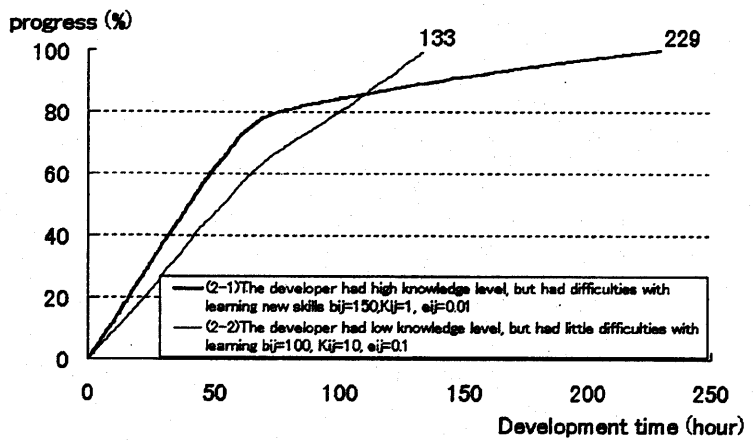

Figure 29. Result of Case2

The consequence of these scenarios is shown in Figure 29. In the result, at the beginning developer(2-1) makes better progress than developer(2-2). But when the progress is at about 80% of the progress, developer(2-1)'s productivity reduces, and when the progress is about at 85%, developer(2-2) makes a better progress than developer(2-1). Finally, developer(2-1)'s effort (development time) is about 1.7 times as much as developer(2-2)'s effort. If the result can be compared to the actual effort of learning of a programming language, an effect of learning language's theories can be seen. Whether a new but similar programming language should be used in a project or not can also be seen.

Case3: A developer does two activities, each activity has a different distribution of knowledge levels.

Software development consists of various kinds of activities. Coding needs knowledge of narrow limits, relatively, while testing needs knowledge of wide limits. Because testing consists of various kinds of activities: creating test data, setting up the test environments, executing, analyzing the test's results, and debugging. These activities have distinct goals, methods, and procedures.

The proposed model has shown its ability to indicate the different distribution of the knowledge level. Three scenarios are given as below:
(3-1)Activity needed knowledge of wide limits.($\sigma = 120$)

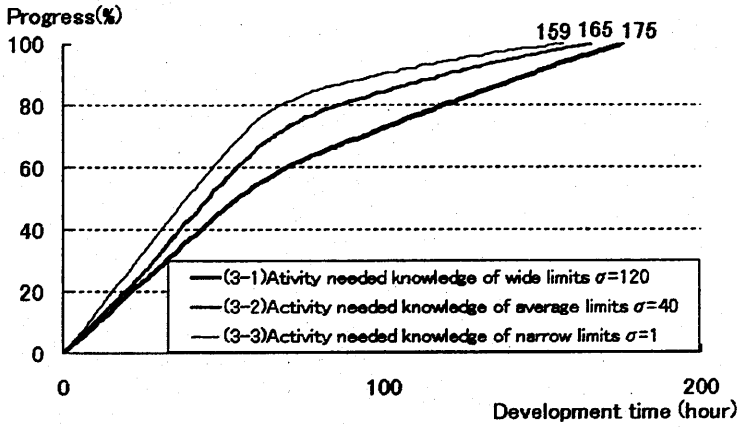

Figure 30. Result of Case3

(3-2)Activity needed knowledge of average limits.($\sigma = 40$)
(3-3)Activity needed knowledge of narrow limits.($\sigma = 1$)

Figure 30 depicts the result of these scenarios. The activity which needed knowledge of wide limits takes more effort than the activity which needs knowledge of narrow limits. But the disparity of their total efforts is very small, as well as the disparity of the progress. As long as the distribution of knowledge is a normal distribution, the difference in distributions of knowledge level will not have a great impact on the development effort and the progress.

Case4: A developer performs two activities, each activity requires a different level of accuracy.

On command based user interface such as UNIX system, if a developer knows the correct commands, his/her productivity will be high. If the developer doesn't know the correct commands, his/her productivity will not only be low, but he/she can't even perform the activity. However, on GUI based user interface such as X-window system, if the developer doesn't know the correct operation, he/she can still complete the task by using the GUI and so on. Under such circumstances, developer's productivity is low, because he/she may need to try the same operation such as clicking menus in searching for the correct one.

The proposed model has shown its capability to differ between the activities required different levels of accuracy. Two scenarios are as follows:

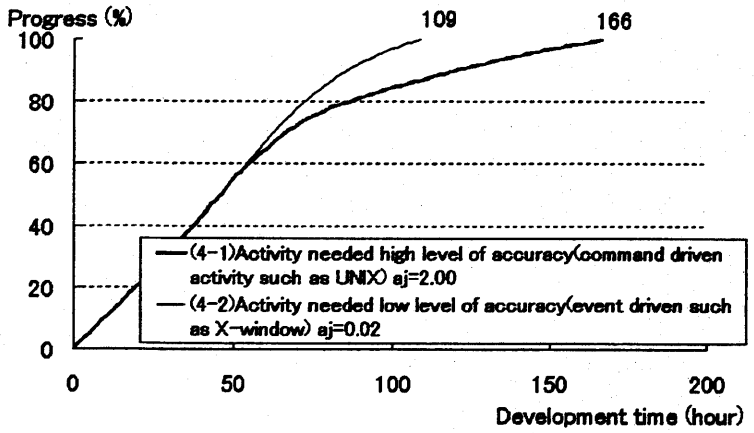

Figure 31. Result of Case4

(4-1) Activity needed high level of accuracy. ($a_j = 2.00$)
(4-2) Activity needed low level of accuracy.($a_j = 0.02$)

The result of the two scenarios is shown in Figure 31. In scenario(4-1), the developer's productivity is reduced in the later half of the development period sharply. The activities in which the developer doesn't have a high level of accuracy are left over in the later half of development period. A little shortage in the developer's knowledge which has been left over reduce the productivity sharply.

Alternatively in scenario(4-2), the developer's productivity is almost invariable during the development period. The productivity is always kept at about half of the maximum productivity under any needed to knowledge levels. Because the knowledge needed to execute the activity is not accurate, the productivity is maximal or 0 rarely during development period.

Current simulation models for software development don't take into account the level of accuracy, as conventional tools always required high level of accuracy. But new tools and development environments of software require low level of accuracy for execution, because they will not only have GUI, but they will have some functions such as online help, navigations and so on. Therefore we need a model to consider the level of accuracy. The proposed model will be able to apply to real projects widely, since the model has the ability to show the gap of accuracy of knowledge in recent tools and development environments, and the

development progress which is influenced by the gap of accuracy of knowledge.

4.3 Customization and evaluation

Although the proposed model is very useful to estimate progress and development periods of projects where developers have to learn new technology, industrial managers may not apply the model to their projects. We thought of two reasons why the managers can not apply the model. The first reason is that managers are required good understanding of the model, that is, the difficulty of assigning values to the parameters of the model. The second reason is that managers don't know how reliable and accurate the results of the simulator are. This section presents a way of applying our proposed model to industry using empirical data. To apply the model to industry, we customize the model for the first reason, and evaluate the model for the second reason.

4.3.1 Approach

There are two popular approaches for evaluation and customization of such models. One is usage of data which was collected from real life completed projects. Another is usage of data which was collected from experiments in laboratories. However, both approaches include significant problems. In the first approach, the data collected from completed projects is influenced by unexpected events such as changes in requirements and hardware problems. The data influenced by the events will confuse us in the evaluation and customization of information or steps. In the second approach, the scale of the experiment is much smaller than real life projects. If we evaluate the model using the data collected from the small scale experiments, we will not be able to evaluate whether the model will compute accurate development periods of large scale projects or not.

Therefore, the new approach was chosen for the evaluation and customization of the model. Because the people who determine the usefulness of the model are industrial managers, we request the managers to cooperate to customize and evaluate the model. The model is customized and evaluated using empirical data which has been collected from the managers.

With the use of virtual projects, the customization and evaluations of the model becomes feasible. Virtual projects are similar to real projects. A virtual

project includes virtual developers who could be in any projects, and virtual activities that could be occurred in any projects. A different point is that virtual projects are not executed actually. The non-execution of the virtual projects is convenient for customizing and evaluating the model. As virtual projects don't involve real budget, time and human resources, the virtual projects could be set as same as industrial projects which managers always control. Moreover,virtual projects can involve intentionally specific problems that are taken up in the model. The proposed simulation model focuses on developers' learning. Assuming that the virtual developers need to learn new technology, the model allows us to simulate their progress and show their learning curves.

4.3.2 Customization

A purpose of the customization is to allow managers to give values to the parameters of the model easily. For this purpose, equations converting answers of questions to the parameters' values are generated.

The model has 8 parameters. Parameters related to activities are as follows:

a_j : level of accuracy needed to execute an activity j

w_j : total amount of activity j

μ : average of θ(the required knowledge level to execute primitive activities)

σ: distribution of θ(the required knowledge level to execute primitive activities)

Parameters related to developers are as follows:

C_{ij}: maximum of developer i's productivity

b_{ij}: knowledge level of developer i

K_{ij}: maximum quantity of gain to knowledge of developer i by executing an activity j

E_{ij}: developer i's efficiency of gain to knowledge by executing an activity j

To make discussion simple, equations of 4 parameters related to developers are generated. The values of the other parameters related to activities are given beforehand. However, only parameter w_j (total amount of activity) becomes a target of making extended equations(See step1-3). The unit of the parameter w_j

is important in the customization. The reason is that there is serious relationship between the unit of the parameter w_j and the units of the parameter C_{ij}. If the unit of the parameter w_j is "Function Point" and the unit of the parameter C_{ij} is "LOC/Month", the results of the simulations will be nonsense. The conversion equations are generated according to 3 steps.

Step1-1 Designing questions

According to each parameter, a corresponding questionnaire about virtual projects is made. When designing these questionnaires, we tried to avoid the use of unfamiliar technical terminology and keep managers away from the task of tedious calculation. The questions of the parameters (C_{ij}, b_{ij} ,K_{ij}, E_{ij}) are shown as follows:

- C_{ij}: maximum productivity
  "Assuming that the developer performs with his highest level of efficiency, what is his maximum productivity?"

- b_{ij}: knowledge level of developer
  "How much does the developer know about this new technology? Please answer using percentage (from 0% to 100%)."

- K_{ij}: maximum quantity of gain to knowledge of developer by executing an activity
  "How many days does the developer need to master the new technology?"

- E_{ij} : developer's efficiency of gain to knowledge by executing an activity
  "Assuming the developer gives up executing the activity, how difficult is it for him to execute the activity? That is, what is the difference between his knowledge and the required knowledge to complete the activity? Please answer using percentage (from 0% to 100%)."

For some parameters, such as maximum productivity C_{ij}, it is easy to derive questions directly from them. In C_{ij} case, as managers are familiar with "productivity," the question is directly derived from the parameter. However, for some parameters, such as maximum quantity of gain to knowledge K_{ij}, there

is no direct relationship between the parameters and the questions. There is a need to devise the questions with commonly known terminology. In K_{ij} case, as experienced managers often consider time to learn a new technology in common practice, the question is to ask the time to learn it.

Step1-2 Making basic equations

The theory of making basic equations is the same as designing questions. When there is a direct relationship between the parameter and question, the parameter is assigned value of the answer. In the other relationship, the parameter is assigned a reciprocal of the value of the answer. The basic equations of the parameters (C_{ij}, b_{ij} ,K_{ij}, E_{ij}) are shown as follows:

- C_{ij} = value of the answer.
- b_{ij} = value of the answer.
- K_{ij} = 1/(value of the answer)
- E_{ij} = 1/(value of the answer)

Step1-3 Making extended equations

However, with the above basic equations, results of the simulation are very different from the results of managers' estimation. This is because the model does

Table 2. Estimation and results of simulation with "all revised constants = 1"

Virtual ProjectNo.	Virtual project1			Virtual project2		
Manager No.	1	2	3	1	2	3
Managers' estimation(Month)	7.5	5.8	6.7	2.5	2.0	3.0
Results of simulation(Month)	1.19	1.18	1.16	0.8	0.86	1.38
Gap of two periods(%)	630.2	491.5	577.6	312.5	232.6	217.4

Table 3. Results of tuning for revised constants of extended equations in step3

Combination No.	Revised constants for w_j	b_{ij}	C_{ij}	K_{ij}	E_{ij}	Average of gap(%)
1	1	1	1	1	1	70.9
88	15	1	1	31.25	6.50	130.9
89	15	1	1	31.25	6.75	119.8
90	15	1	1	31.25	7.00	100.64
91	15	1	1	31.25	7.25	96.4
92	15	1	1	31.25	7.50	86.3

not consider the units of the parameters. In the above step1-1 and step1-2, the units of the parameters are "hour" and "percentage". We don't know that the model can compute accurate periods using "hour" and "percentage". The basic equations have to be tuned up for closer, accurate results. There are various ways for the tune up. A simple approach is chosen. The extended equation is as follows:

Extended equation = Basic equation * Revised constant

The determination of the above revised constants is a result of a series of tests. The tests involve comparing managers' estimation with the result of simulation using different trial constants. When the difference is minimal, the constants are optimal. The optimal values are chosen as revised constants. These tests are based on the assumption that experienced managers always estimate accurate development periods. Therefore results of the simulation should be same as the managers' estimation.

The extended equations of the parameters (w_j, C_{ij}, b_{ij} ,K_{ij}, E_{ij}) are shown as follows:

- w_j = LOC of program * **15**
- C_{ij} = value of the answer * **1**
- b_{ij} = value of the answer * **1**
- K_{ij} = 1/(value of the answer) * **31.25**

- E_{ij} = 1/(value of the answer) * **7.00**

The bold numerical characters in the above equations are the revised constants. In this step, three experienced managers estimated development periods of two virtual projects. They answered the questions made in step1-1. Table 2 shows the managers' estimations and the results of simulations in which all values of the revised constants were set to 1. The gaps of two periods in Table 2 are large (form 217.4% to 630.2%) because the model doesn't consider the unit of the parameters. Table 3 shows the results of the tuning for generating the revised constants. The revised constant for w_j is increased from 1 to 50. The quantity of the increment is 1 on each trial test. The other revised constants for C_{ij}, b_{ij} K_{ij} and E_{ij} are increased from 1 to 50.0. The quantity of the increments is 0.25. On Combination No.90 in Table3, the gap of two development periods was the smallest: only 100.64%. Therefore, the revised constants were determined on the above equations.

4.3.3 Evaluation

A purpose of the evaluation is that we give managers an assurance that the model can compute development periods as accurately as experienced managers' estimation. The customized model with the extended equations is evaluated. Therefore the result of this evaluation means an assurance of the customized model and not the original one. And that, experienced, independent managers who did not generate the extended equations evaluate the customized model. Procedures of the evaluation consist of 4 steps.

Step2-1 Setting up virtual projects

Virtual projects are set up to reflect the features of the model. In this case, virtual projects are set up in a situation that developers have to learn new technology. Because there are various domains which managers are good at, six virtual projects were prepared. For example, in Virtual Project No.1, a virtual developer makes a production management system in C language on WindowsNT. The virtual developer has only experience of programming in COBOL. The virtual developer has to learn C language in the virtual project. Some parts of Virtual Project No.1 are described in Table 4.

Table 4. Virtual Project No.1

Software	Business application for production management. Client and Server system on WindowsNT.
Activity	Input Product into the activity:Document of system design(50 pages). Output product from the activity:C programs (are completed unit test and integrated test). Activities: –Structural design for programs. –Detail design of function. –C programming. –Unit test for Function. –Integrated test for programs.
Developer	NAME: Taro Yamada *His educational background: He has a bachelor of Information Technology degree and majored in Software Engineering. *His education after graduation: He had given training for 3 months. The contents of the training are: an introduction of C language(for a month), an introduction of JAVA language(for a month) and a fundamental knowledge of Software Engineering(for a month). *Experience of software development: (1)Banking system (MainFrame, 3 years) —Design of program, programming, unit, integrated, total test —Document for program design: 240 pages —COBOL program:54KLOC —Document for Integrated test: 100pages —Document for total test: 300pages COBOL —Productivity of programming(COBOL) : 5KLOC/month —Productivity of writing documents: 17.8 pages/month Interview to the developer: *Are you interested in new technology? —Yes, I'm interested in the latest technology. *Do you have confidence in learning new technology ? —I don't have enough confidence.

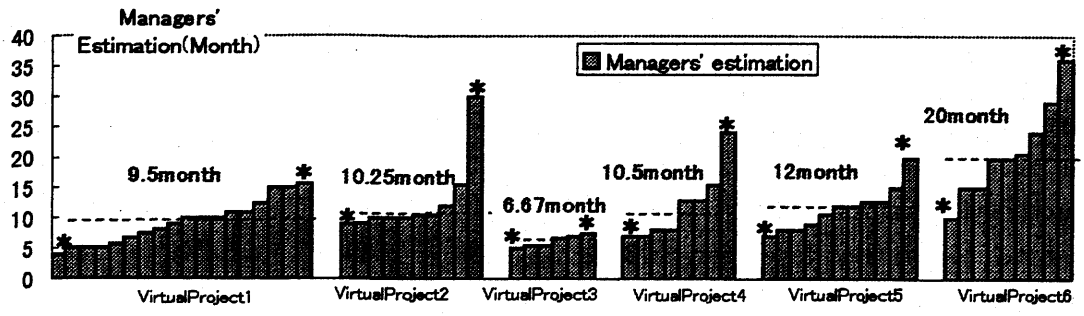

Figure 32. Managers' estimation

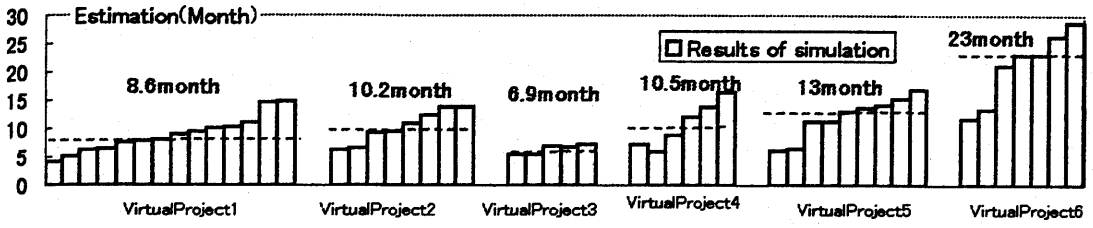

Figure 33. Results of simulation with managers' responses

Step2-2 Acquisition of managers' knowledge

Managers who are not involved in the making of the extended equations are asked to give estimation about virtual projects. Table 5 presents five managers' responses to the questions about Virtual Project No.1. For example, Manager No.1 estimated that the virtual developer takes 10 months to execute the virtual activity. And the manager decided that the developer's maximum productivity using the new technology is 2.0 KLOC/month, the developer's knowledge about C language is 20%, the developer needs 90 days to master C language, and the developer gives up executing the activity which is 20% more difficult than his/her knowledge. Figure 32 shows 62 estimations of the managers. The broken lines in Figure 32 mean medians of the estimations of virtual projects. The median of Virtual Project No.1 is 9.5 months, the median of Virtual Project No.2 is 10.25 months and so on.

Step2-3 Simulation using managers' knowledge

The responses from the managers become the input values of the corresponding parameters. The model then simulates development periods with the extended

Table 5. Responses about Virtual Project No.1

Manager No.	1	2	3	4	5
What is developer's maximum productivity? (KLOC/month)	2.0	1.5	2.0	1.0	1.5
How much does the developer know about C language?	20%	10%	30%	10%	40%
How many days does the developer need to master C language?	90days	60days	200days	100days	100days
What is difference between his knowledge and the required knowledge to execute ?	20%	30%	30%	30%	30%
Estimated development period(Month)	10	11	15	15	10

equations. Figure 33 shows results of simulation with managers' responses. The median of Virtual Project No.1 is 8.6 months, the median of Virtual Project No.1 is 10.2 months and so on.

Step2-4 Analysis of a gap of two development periods

The simulated development periods (from Step 2-3) are compared with the managers' estimations (from Step 2-2). The gap between the two periods is analyzed.

At first, a gap between each manager's estimation and each result of simulation is analyzed. The results of the analysis are shown in Figure 34. In the graph, the X-axis is the estimated development periods; the Y-axis is the simulated development periods. If the plots in the graph are on the dash line, the gap of two development periods is very narrow; that is, the simulated period is very close to the managers' estimation. Every manager's estimation of a virtual project is different. Some tend to estimate a shorter development period; others tend to estimate a longer development period. This is because of the difference in everyone's experience. As a consequence the estimated development periods of a virtual project are not same. Therefore there are various gaps between the estimated development periods and the simulated development periods. Assuming that the distribution of the gaps of the two development periods is a normal distribution, the criteria to judge how small the gap is based on X^2 test (chi-square

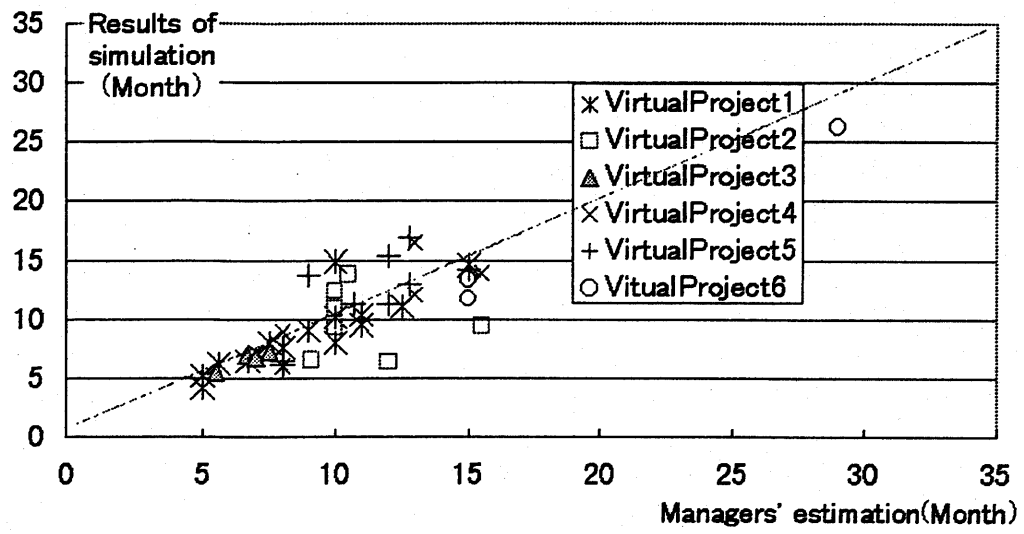

Figure 34. Result of evaluation

test)[29]. As a result of X^2 test (chi-square test), the value of X^2 is 23.98. When the degree of freedom is 50, the value of $X^2_{0.01}$ is 29.71. That is, the results of the simulations by the customized model fit managers' estimation at 1% level of statistical significance.

Next, a gap between the median of manager's estimation and the median of result of simulation is analyzed on each virtual project. For example, in Virtual Project No.1, the median of managers' estimation is 9.5 months, the median of results of simulation is 8.6 months(See Table 6). The gap between the median of managers' estimation and the results of simulation is 0.9 months. As a result of X^2test (chi-square test) of the gaps of the medians, the value of X^2 is 0.64. When the degree of freedom is 5, the value of $X^2_{0.05}$ is 0.71. That is, from the point of view of median on each virtual project, the results of the simulations fit managers' estimation at 5% level of statistical significance. Therefore we give managers an assurance that the customized model can compute development periods as accurately as experienced managers' estimation.

Table 6. Results of estimation

Virtual Project No.	1	2	3	4	5	6
Medians of managers' estimations(month)	9.5	10.25	6.67	10.5	12.0	20
Medians of the simulated estimations(month)	8.6	10.2	6.9	10.5	13.0	23.0

4.4 Application

4.4.1 A Simulator

We provide a new software development simulator which is based on the customized and evaluated model to industrial project managers (See Figure 35). As the simulator is based on WWW, anyone can simulate a progress of his/her projects from in different parts of the world. To help users understand results of the simulation, the system produced three graphs. (See Figure 35). The progress graph shows how the project progresses. The learning curve graph shows the developer's learning curve. The activity graph shows how the developer finishes the activities.

4.4.2 Case study

We applied the simulator to six cases in which the development progress is influenced by the developers' learning. The parameters related to activities are given beforehand. The parameter μ (average of θ) is given 50, the parameter σ (distribution of θ) is given 40, the parameter a_j (level of accuracy needed to execute an activity) is given 10.

Case1: Total amount of an activity(w_j) is 10KLOC(line of code)

It assume that the activity includes design of programs, programming, program tests and combination test.

Case1-1: Conventional technology

A developer makes software using conventional technology such as C language and structured design/programming methodology. He/she has enough knowledge of

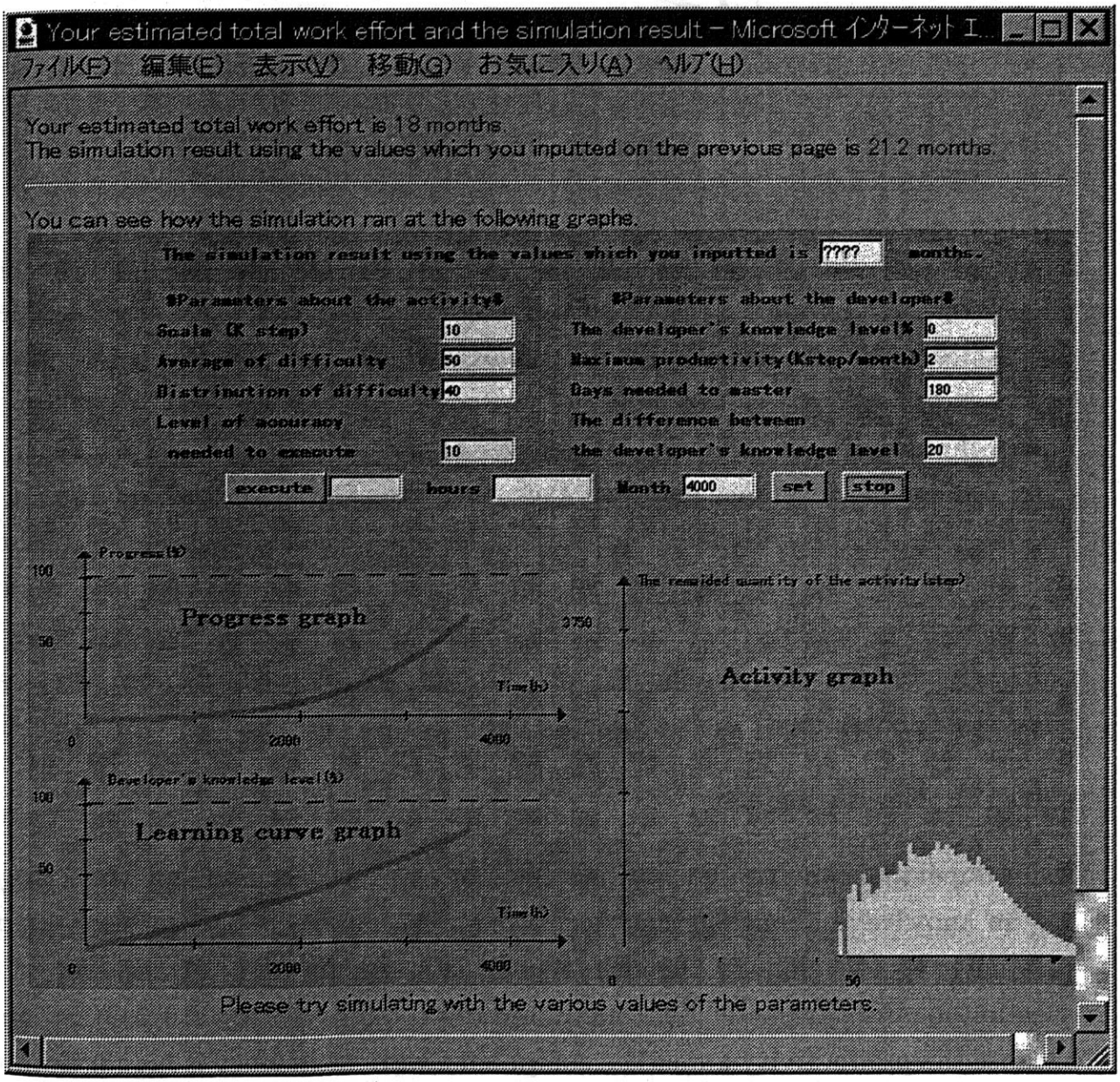

Figure 35. Simulator

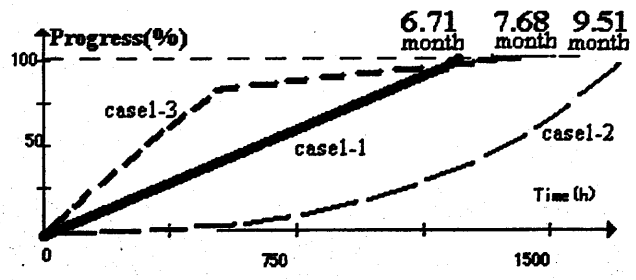

Figure 36. Result of Case1

the technology. An answer of the question to the parameter b_{ij}(developer's knowledge level) is maximum (100%). An answer of the question for the parameter C_{ij}(maximum of productivity) is 1.5 KLOC/Month. An answer of the question to the parameter K_{ij} ("How many days does the developer need to master the new technology?") is 100 days. An answer of the question to the parameter E_{ij} ("Assuming the developer gives up executing the activity, how difficult is it for him to execute the activity?")is 30%. The line "case1-1" in Figure 36 shows the simulation result of Case1-1. We find that the developer takes 6.71 months to execute the activity in Case1-1. Moreover the progress of Case1-1 goes up straightly. In this case, managers can estimate and/or predict the progress and development periods of their projects easily.

Case1-2: New technology

A developer makes software using new technology such as component-ware. He/she has no knowledge about the new technology. An answer of the question for the parameter b_{ij} (developer's knowledge level) is minimum (0%). An answer of the question for the parameter C_{ij} (maximum of productivity) is 3.0 KLOC/Month, because the productivity under the component-ware technology is higher than the productivity under the conventional technology. The other answers are as same as answers of Case1-1. The curve "case1-2" in Figure 36 shows the simulation result of Case1-2. The developer takes 9.51 months to execute the activity. In the early stage of the project, the productivity of Case1-2 is low, because the developer needs to learn the new technology. However, in the middle stage, the productivity becomes higher and higher. In this case, if managers can not predict

the variation of the productivity in the middle stage, they may misunderstand that their projects will be behind schedule in the early stage of the project.

Case1-3: Difficult technology

A developer makes software using difficult technology such as object-oriented method. He/she has the main part of knowledge of the technology. An answer of the question to the parameter b_{ij} (developer's knowledge level) is 70%. If the developer masters the technology, the developer's productivity will be very high. The answer of the question to the parameter C_{ij} (maximum of productivity) is 5.0 KLOC/Month, because the developer can reuse many objects which have been developed in object-oriented technology. However, it needs long time for the developer to master the technology. An answer of the question to the parameter K_{ij} ("How many days does the developer need to master the new technology?") is 300 days which is three times as much as the values of the parameter K_{ij} of Case1-1 and Case1-2. Moreover, it is difficult for the developer to learn the technology during the execution of the activity. For example, when the required knowledge level to execute the activity is even a little higher than the developer's knowledge level, the developer must give up executing the activity. An answer of the question to the parameter E_{ij} ("Assuming the developer gives up executing the activity, how difficult is it for him to execute the activity?") is 10%. 10% is less than the values(30%) of E_{ij} in Case1-1 and Case1-2.

The curve "case1-3" in Figure 36 shows the result of Case1-3. The developer takes 7.68 months to execute the activity. In the early stage of the project, the productivity of Case1-3 is very high. However, in the middle stage, the productivity becomes lower and lower, because the developer can gain little knowledge during the execution of the activity. In this case, if managers can not predict the variation of the productivity in making schedule, their projects will be late in delivering their software.

In Case1, because the scale of the software is small relatively, the developer in Case1-1(usage of conventional technology) can complete the activity in the shortest time(6.71 months). The reason is that time to learn technology is not required in spite of the low productivity of conventional technology. That is, in small scale software, if manager would like to complete their project as soon as possible, they should choose the conventional technology in their projects.

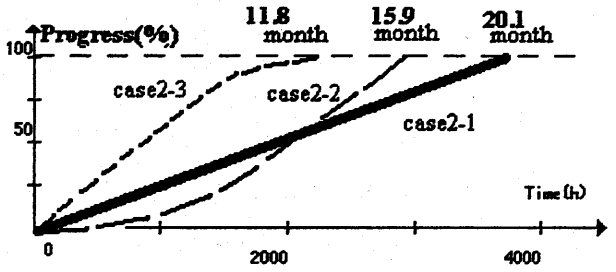

Figure 37. Result of Case2

Case2: Total amount of an activity(w_j) is 30KLOC(line of code)

In Case2, the total amount of the activity is larger than the total amount of the activity in Case1. The other conditions of Case2 are same as the condition of Case1. In addition, the values of parameters of Case2-1 are same as the values of parameters of Case1-1, the values of parameters of Case2-2 are same as the values of parameters of Case1-2, the values of parameters of Case2-3 are same as the values of parameters of Case1-3. Results of Case2 are showed in Figure 37. A developer of Case2-1(convention technology) takes 20.1 months, a developer of Case2-2(new technology) takes 15.9 months and a developer of Case2-3(difficult technology) takes 11.8 months to complete the activity.

The most different point between the results of Case2 and the results of Case1 is that the developer of Case2-1(usage of conventional technology) needs the longest time(20.1 months) to complete the activity in Case2. As the productivity of Case2-1(conventional technology) is lower than the productivities under the other technologies, the influence of the low productivity on the completion date is more than influence of the time to learn the technology. The time to learn the technology in Case2-2 and Case2-3 is as same as the time to learn in Case1-2 and Case1-3, respectively. On the other hand, total work time of Case2 becomes longer than total work time of Case1, because the scale of the software of Case2 is larger than it of Case1. Therefore, the ratio of the time to learn the technology in the total work time of Case2 becomes low. As a result, the influence of the time to learn the technology on the completion date is low.

Our simulator can make clear the gap of small project and large project in using various technologies. That is, if the software is large enough, manager should choose a new technology in which developers can achieve high productivity in spite of needs to learn it. If the software is small, managers should choose a conventional technology which does not require the time to learn it. The simulator is useful when managers decide which technology should be chosen in various scale projects.

4.5 Conclusion

In this chapter, a new simulation model is proposed. The model is based on the developer's learning during software development. Development progress can be estimated in line with our proposed model. In order to prove the correctness of our proposed model, software development in four cases in which the characteristic of activities and developers may influence the development progress has been simulated. As a result, our proposed model could be applicable to real life projects, and the presented model has an ability to simulate various situations of development progress.

In addition, the proposed model is customized and evaluated because industry project managers can simulate their project progress in the model. The customized and evaluated model is implemented as a project simulator. The managers can estimate their progress and development periods with only answering questions about their project in the simulator. Six case studies when developers use conventional or new technology are also shown. Gaps of progress in different technologies have been clear. As a result, the simulator helps the manager to decide whether the new technology should be used in his/her project or not.

In the future, the proposed model will be refined according to the following topics.

(i) Developer's knowledge is divided into static knowledge and dynamic knowledge. Static knowledge means whether the developers know it or not. Dynamic knowledge means whether the developers can use it or not.

(ii) Using the model, a clear order of primitive activities which will be executed will be defined as a project's strategy.

(iii) Managers evaluate not only the development periods, but the halfway progress in the simulation.
(iv) Some developers will not finish an activity because of limits of learning.

5. The Project Planning Prototype

This chapter presents a new project planning prototype tool based on the proposed framework and simulation model. The prototype was implemented in Visual Basic, and three package softwares: Visio[33], MS Access[21] and MS Project[22]. Using the prototype, managers can make project plans based on these proposed methods easily, and the prototype helps the managers decide which technology should be chosen in their projects.

Figure 38 shows an outline of the prototype which consists of four components: a generation of software process description, an assignment of resources, a simulation based on the proposed model, and a generation of project plans using an activity bar chart.

5.1 The four components

We describe the components of the prototype using a situation which is illustrated in Figure 39 and Figure 40.

(1)The generation of software process description

The software process description can be generated automatically using our already proposed framework[11]. In this framework, if a data flow diagram(DFD) of development method is given on Visio, a software process description including relevant milestones will be automatically generated as Visio's charts.

(2)The assignment of resources

The assignment of resources could be entered by a project manager into the prototype. The items which the manager should enter are the parameters of the proposed model. In Figure 39, there are six activities which are named *PHASE1* to *PHASE6* in *Activity Table*, these activities have been already determined by the generation of software process description.

Each activity has a *Size* which is entered by the manager in the *Estimation Table*. *Size* is total amount of the activity w_j of the Activity model(See Equation(1)). In this project, there are five developers: Developer1(Noriko), Developer2(Satiko), Developer3(Hiroko), Developer4(Kumiko), and Developer5(Reiko) in the *Developer Table*. Each developer has an engineer rank *Rank_Name*. The engineer rank consists of four parameters of the proposed model: the developer's

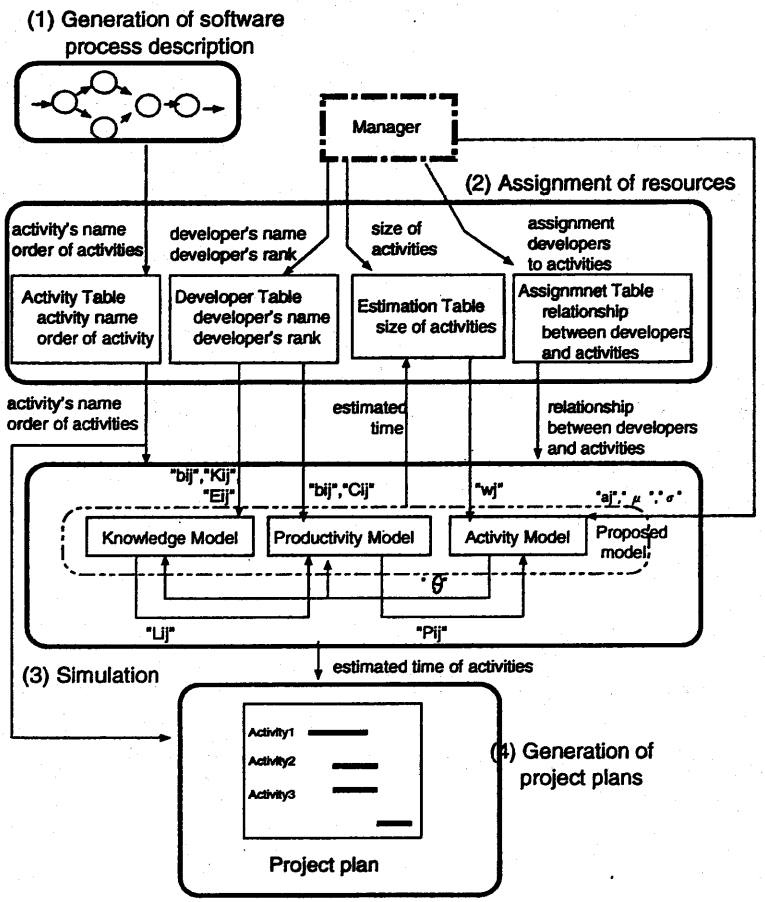

Figure 38. The outline of Project planning prototype

knowledge level b_{ij}, the developer's maximal productivity C_{ij}, the developer's maximal quantity of gain to knowledge K_{ij}, and the developer's efficiency of gain to knowledge E_{ij}. Table 7 shows the values of the engineer's ranks. The manager has to decide the developers' ranks. Finally, the manager assigns the developers to the activities in the *Assignment Table.*

(3) The simulation

The simulation starts after assignments of resources. The estimated time of each activity is determined by the entered values, using the proposed model. Values of the field *Estimate_Time* of *Estimation Table* in Figure 39 are the results of this simulation. It takes thirteen days to complete the activity *PHASE1* by three developers; Developer1(Noriko), Developer2(Satiko),and Developer3(Hiroko).

(4)The generation of project plans

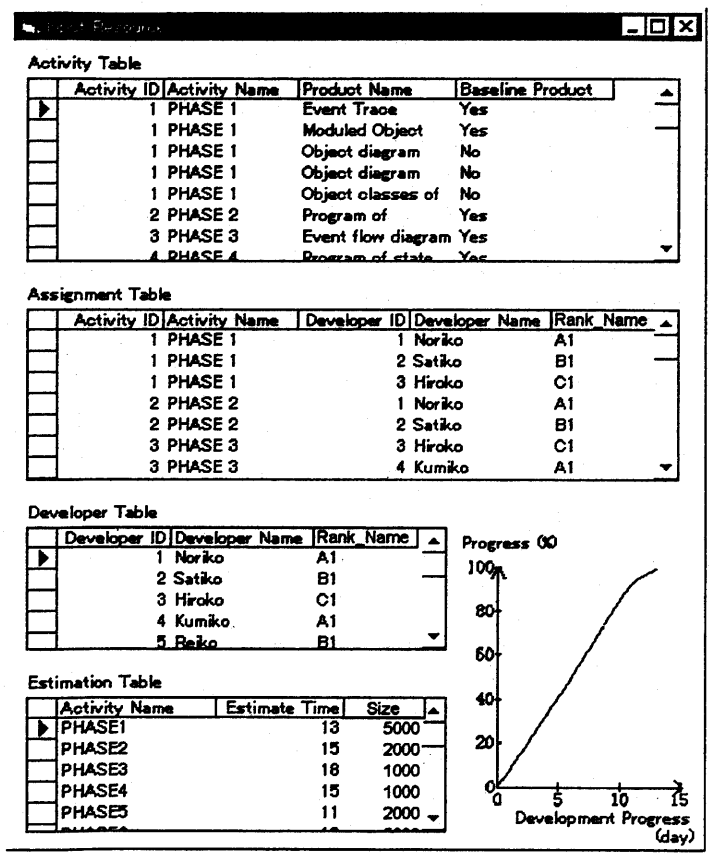

Figure 39. Assignment of resources

Activity bar charts are made in Microsoft Project[22] (See Figure 40). They have the started date and the completed date of each activity. The started date and the completed date of each activity are calculated by two factors. One is the estimated time of the activities, the other is the order of activities which have been already determined by the generation of software process description[9]. In this case, The stated date of the activity *PHASE1* is the 25th of August, the completed date of the activity *PHASE1* is the 10th of Sep. 1997. The due date of the project is the 4th of Nov. 1997.

5.2 A feature of the prototype

One of the important implications of the prototype is that, it indicates if a new and unfamiliar software development environment is applied, the project will have a high possibility to be late, in spite of the constancy of the quantity of the activities and the assignment of resources. This is due to the fact that the acquisition and application of the new environment is time consuming[2][35].

The prototype has demonstrated an ability to reschedule software projects. Usually the developer's knowledge level in the new software development environment is lower than the developer's knowledge level in the conventional software development environment. For example, under the new software developing environment, Developer1(Noriko)'s rank changes from *A1* to *A2*. The value of b_{ij} of rank *A1* is 78 %, the value of b_{ij} of rank *A2* is 58%. The other parameters such as C_{ij}, K_{ij}, and E_{ij} have the same values.(See Table 7). A reduction of b_{ij} means

Table 7. The values of the engineer rank

Rank_Name	developer's knowledge level(%)	maximum productivity (KLOC/ Month)	days to master (days)	at giving up, gap of two knowledge(%)
A1	78	3.0	42	49.0
A2	58	3.0	42	49.0
B1	58	3.0	69	24.5
B2	38	3.0	69	24.5
C1	38	2.0	83	21.0
C2	18	2.0	83	21.0

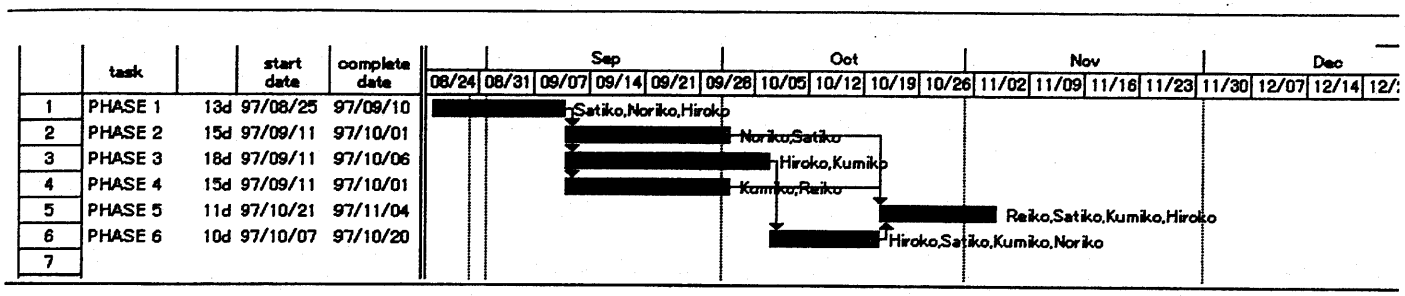

Figure 40. Schedule for software development

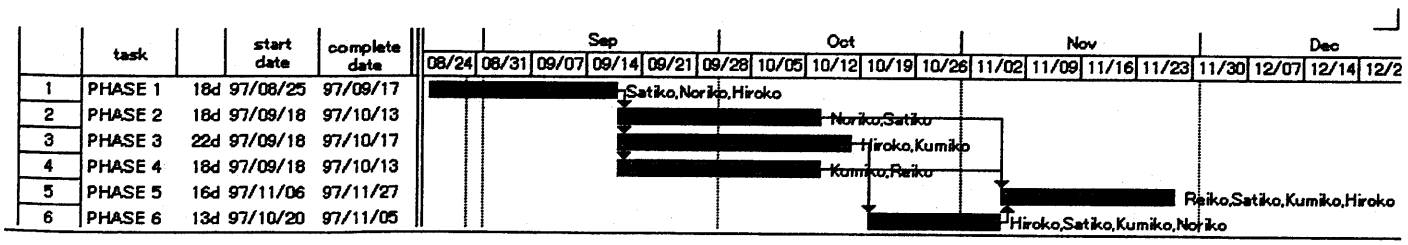

Figure 41. Re-schedule for software development

that Developer1(Noriko) is not familiar with the new software developing environment. Similar phenomenon applies to other developers. Developer2(Satiko)'s rank changes from *B1* to *B2*, Developer3(Hiroko)'s rank changes from *C1* to *C2*, Developer4(Kumiko)'s rank changes from *A1* to *A2*, Developer5(Reiko)'s rank changes from *B1* to *B2*. The developers' b_{ij} are decreased to 20% respectively.

From Figure 41, we could see the result of such a simulation. It takes eighteen days to do activity *PHASE1*, which is a week more than the estimated time "thirteen days" which was calculated in Figure 40. The due date of the project is the 27th of November, it is twenty three days behind the calculated due date. Therefore, if the manager has to complete the project by the 20th of November, he/she should not use the new software environment in the project. The prototype can be a good indicator to see whether a new software developing environment should be used.

5.3 Conclusion

A new project planning prototype tool has been provided. The tool generates project plans as bar-charts. We can compare a plan in conventional development methods with a plan in new development methods. The tool helps project managers decide which development methods are chosen in their projects.

6. Conclusion and Future Research

6.1 Summary of major results

This thesis has proposed a useful method for software development projects management. A basic reason of the proposed method is that the planning in the management should be adaptable to software development paradigm shifts. If managers make a plan using conventional phases without considering activities of the new methodologies such as object-oriented development methodologies, managers will not be able to comprehend actual progress of the project in their unsuitable plan. The new framework can generate an object-oriented development process with relevant milestones. The generated process in the framework identifies management activities and phases that consist of development activities of object-oriented development methods. Moreover, the process defines the sequence of these phases and these management activities. If project managers make their plans using the clarified phases, activities and the sequence in the generated object-oriented process, the project plans will be more proper than plans based on conventional phases. The new simulation model is useful for estimating development periods of project plans using new development environments such as CASE tools. Especially, when developers of the projects have to be familiar with new environments and technologies, the simulation model can calculate more accurate development periods than the calculated periods by fixed productivity.

In chapter 2, project management activities were described. Especially, in planning, importance of generating the software process and difficulty of estimating resources were explained. Moreover, related works to methods of process generation and simulation models were shown

Chapter 3 describes the new framework which gives us a guideline for generating a software process with relevant milestones for object-oriented development methods. The framework identifies phases which consists of development activities of object-oriented development methods. Because the identified phases include all activities which update the same products, the reviews at the end of the phases will be good milestones. Managers will be able to comprehend their projects' progress using the milestones. In addition, baseline products are identified in the framework. The identification of the baseline products will make

reviews' low cost, because members of the reviews only judge the identified baseline products among many products in projects. Application of the framework is not limited only for object-oriented development methods. The framework has potential ability to be able to apply to the other development methods which define clear relationship among activities and products.

Chapter 4 describes the new simulation model which can estimate development periods by considering variations of productivity. In the new development environment, developers' productivity is low at the beginning of the project in spite of high productivity of the new development environment, because it takes much time to become familiar with the new environment. The model can calculate the growth of a developers productivity as the project progresses. In addition, the model can show a developers' learning curve during the project. To apply the model to industry, the model was customized and evaluated. In the customization, five equations were added to the model. The equations could calculate the values of the model's parameters based on the answers of questions. Therefore, the model was convenient and easy for project managers to use. In the evaluation, results of simulation were compared with experienced managers' estimations. As a result, simulated periods fit managers' estimated periods at 5% level of statistical significance. Therefore, results of the simulation using the model were as accurate as experienced managers' estimations.

Chapter 5 presented the project planning prototype based on the proposed method and model. The prototype tool could generate project plans as bar charts. Managers compared two plans: a plan using a conventional methodology and/or conventional environment and a plan using a new methodology and/or new environment. Usefulness of the prototype tool in deciding which methodology and environment should be chosen was shown.

6.2 Future works

Taking a broad view is important in software development because many elements influence software development projects. If you manage a software development project shortsightedly, you will be not able to lead your project to success. We think that a software development project mainly consists of three elements: human, methodology and environment. Human means developers, managers and

customers. Methodology means theories of how to develop software and how to use it. The environment means hardware such as mainframe computers or networked personal computers and software such as CASE tools and programming languages. For example, human as a developer executes activity which is defined in a methodology on a personal computer in a development environment. These elements influence each other. If only one element is focused in software engineering research area, the results of the research may not be useful for software development projects. It is important for us to clarify the relationship among these elements.

The proposed method considers the relationship among these elements. For example, the estimating model has taken account of developers' learning. The learning is an important factor of the human element. Moreover, the developers' learning is caused by the shifts of the methodologies and environments. The model has clarified a relationship among the methodology element, the environment element and the human element .

Therefore, we have to focus more on the relationships among these elements. For example, the influence of a transition from "Intel 8086 CPU" to "Pentium III CPU" on the methodology element should be clarified. Probably, the transition may change how to develop software. That is, developers can display many windows at a time which display programs on a debugger, sample programs in online help and documents on a computer, moreover, the developer copies and pastes parts of sample programs to their own programs on the debugger. The "copy and paste" activity may influence how to develop software. Software development methodologies should be constructed by taking into account the influence of the "copy and paste" activity. In addition, the transition of CPU may influence the human element. For instance, a developer's design ability may become low because "copy and paste" activity of sample programs does not require design ability. The reuse of the sample programs is not only code, but design of the sample programs. This can lower the developers' design ability by influencing parts of the development methodologies.

In the future, a relationship model among these elements will be constructed in development perspective and management perspective. If the relationship among the three elements is clarified, more software development projects will succeed.

The relationship model will give answers to many questions: "why will many software development projects fall into confusion?", "what's wrong?", "who is wrong?" and "which factor is the confusion caused by ?".

Acknowledgements

During the course of this work, I have been fortunate to have received assistance from many individuals. I would deeply like to thank Professor Katsuro Inoue of Software Engineering Laboratory for his precious support and valuable suggestions.

I am also very grateful to Professor Koichi Nishitani of Systems and Control Laboratory for his invaluable comments and helpful suggestions concerning this thesis.

I also wish to thank Associate Professor Ken-ichi Matsumoto of Software Engineering Laboratory for the valuable suggestions and discussions.

I would especially like to thank Vice President Koji Torii for his continuous support, encouragement and guidance for this work.

I would like to give thanks to Associate Professor Hajimu Iida of Information Technology Center for the useful leading and heated discussion.

I would like to express my thanks to Research Associate Kazuyuki Shima of Software Engineering Laboratory for his stimulating discussions and helpful criticism.

I would like to acknowledge also the valuable suggestions of Research Associate Akito Monden of Software Engineering Laboratory in preparing this thesis.

I also wish to express my gratitude Associate Professor Kumiyo Nakakoji of Cognitive Science Laboratory for the instructive suggestions.

I would like to thank to Assistant Professor Singo Takada of Keio University for the careful suggestions on the paper which formed the basis of Chapter 3 of this thesis.

I am also thankful to Professor Dieter Rombach, Dr.Frank Bomarius, Mr.Dietmar Pfahl of Einrichtung Experimentelles Software Engineering, Mr.Raimund L.Feldman and Jurgen Munch of AG Software Engineering Fachbereich Informatik for their insightful comments and valuable discussions on the papers which formed the basis of Chapter 4 of this thesis.

I would like to acknowledge also to the late Mr.Yamanouchi of Software Management Society, Mr.Tohru Matsuodani of NEC Corporation, Mr.Koji Kondo of Sony Corporation and Mr.Hideo Kudo of Nara National College of Technology for the cooperation of collecting responses which were used to evaluate the pro-

posed model in Chapter 4. I also thank to Mr.Akihiro Tamaki, Mr.Makoto Sakai, Mr.Keishi Sakamoto, Mr.Yoshitomi Morisawa for the responses for the evaluation of the model.

I wish to express my gratitude to Mr.Reiken William for his careful reading of a draft of this thesis. His suggestions were very helpful.

Finally, my special thanks are express to Mr.Yoshimi Usui, Mr.Shuji Morisaki, Mr.Yasuhiro Takemura, Mr.Masatake Yamato and Mr.Shohei Morikawa for their help and cooperation. Also, I can never thank my son and my husband enough.

References

[1] Abdel-Hamid T.and Mdnick S.E., "Software Project Dynamics an Integrated Approach," Prentice-Hall, 1991.

[2] Bochenski B., "Implementing Production-quality Client/Server Systems," John Wiley and Sons, 1994.

[3] Booch G., "Object Solutions Managing the Object-Oriented Project," Addison-Wesley, 1996.

[4] Carnegie Mellon University Software Engineering Institute, "The Capability Maturity Model: Guidelines for Improving the Software Process," Addison-Wesley, 1994.

[5] Clapp J.A., "Management Indicators," in Encyclopedia of Software Engineering, J. J. Marciniak, ed., New York, John Wiley and Sons, pages 637-644, 1994.

[6] Jalote P., "A Integrated Approach to Software Engineering," Springer, 1997.

[7] DeMarco T., "Structured Analysis and System Specification," Prentice-Hall, 1979.

[8] DeMarco T., "Controlling Software Projects," Youdon Inc., 1982.

[9] Hanakawa N., Iida H., Matsumoto K. and K. Torii, "A Framework of Generating Software Process including Milestones for Object-Oriented Development Method," *Proceedings of Asia-Pacific Software Engineering Conference*, pages 120–130, 1996.

[10] Hanakawa N., Morisaki S. and Matsumoto K., "A Learning Curve Based Simulation Model for Software Development," *Proceedings of 20th International Conference on Software Engineering*, pages 350–359, 1998.

[11] Hanakawa N., Iida H., Matsumoto K., Torii K., "Generation of Object-Oriented Software Process using Milestones," International Journal of Software Engineering and Knowledge Engineering, Vol.9, No.4, pages445–466, 1999.

[12] Hirayama Y., Mizuno O., Kusumoto S., and T. Kikuno, "Hierarchical project management model for quantitative evaluation of software process," *Proceedings of International Symposium on Software Engineering for the Next Generation*, pages 40–49, 1996.

[13] Humphrey W. S., "A Discipline for Software Engineering," Addison-Wesley, 1995.

[14] Iida H., Eijima J., Yabe S., Matsumoto K. and Torii K., "Simulation model of overlapping development process based on progress of activities," *Proceedings of 1996 Asia-Pacific Software Engineering Conference*, pages 131–138, 1996.

[15] Jacobson I., Christerson M., Jonsson P. and Overgaard G., "Object-Oriented Software Engineering," Addison-Wesley, 1992.

[16] Katayama T., "A hierarchical and functional software process description and its enaction," *Proc. of the 11th International Conference of Software Engineering*, pages 343–353, 1989.

[17] Kellner M. I., "Software process modeling support for management planning and control," *Proc. of the 1st International Conference on Software Process*, pages 8–28, 1993.

[18] Koontz H and O'Donnell C., "Principles of Management," An Analysis of Managerial Functions, 5th Ed., McGraw-Hill Book Co., Inc., New York, 1972.

[19] Kusumoto S., Mizuno O., Kikuno T., Hirayama Y., Takagi Y. and Sakamoto K., "A new software project simulator based on generalized stochastic," *Proceedings of 19th International Conference on Software Engineering*, pages 293–302, 1997.

[20] Lord, F. M., Novick M.R., "Statistical Theory to Mental Test Scores with Contributions by A-Birnbaum," Addison-Wesley, 1968.

[21] Microsoft corp., "Microsoft Office2000 programmer's guide," Microsoft corp., 1999.

[22] Microsoft corp., "Microsoft Project for Windows95 user's guide," Microsoft corp., 1995.

[23] Mochizuki S., Yamauchi A. and Katayama T., "Analyzing and evaluating fundamental design process of checkout system for artificial spacecraft," *Proc. of the Fifth Annual International Computer Software and Applications Conference*, 1991, pages 507–514.

[24] Northrop L.M., "Object-Oriented Development," in Encyclopedia of Software Engineering, Marciniak, J. J. ed., pages 729–737, New York, John Wiley and Sons, 1994.

[25] Putnam L. H., Myers W., "Industrial Strength Software," IEEE Computer Society Press, 1996.

[26] Reifer D.J., "Cost Estimation," in Encyclopedia of Software Engineering, Marciniak, J. J. ed., pages 209–220, New York, John Wiley and Sons, 1994.

[27] Rumbaugh J., Blaha M., Permerlani W., Eddy F. and Loresen W., "Object-Oriented Modeling and Design," Prentice-Hall, Englewood Cliffs, 1991.

[28] Schach S.R., "Software Engineering," Richard D. Irwin Inc. and Aksen Associates Inc., 1990.

[29] Snedecor G.W.and Cochran W.G., "STATISTICAL METHODS," The IOA state university, 1980.

[30] Sommerville I., "Software Engineering," 4th Edition, Addison-Wesley, 1992.

[31] Tyayer A.B. and Pyster R.C., "Major Issues in Software Engineering management," IEEE Transactions of Software Engineering, Vol.7, No.4, pages333–342, 1981.

[32] Thayer R.H., Fairley R., "Project Management," in Encyclopedia of Software Engineering, Marciniak, J. J. ed., pages 900–923, New York, John Wiley and Sons, 1994.

[33] Visio corp., "User's guide Visio 4J," Visio corp., 1995.

[34] van Vliet J.C., "Software Engineering - Principles and Practice," John Wiley and Sons, 1993.

[35] Yourdon E., "Object-Oriented System Design," PTR Prentice-Hall, Englewood Cliffs, 1994.

Appendix

A. Managers' responses of quetions

Meaning of tags of the following response

- NAME:Manager's Name (anonymous).
- バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験: Experience of similar system ("はい" means yes, "いいえ" means no.)
- バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験: Experience of similar environment ("はい" means yes, "いいえ" means no.)
- 管理者の指定言語での開発プロジェクトの参加経験: Experience of programming language("はい" means yes, "いいえ" means no.)
- 管理者の経験年数:Experience period of managing (years)
- 管理者の入力値使用可能？:May I use your input data for analysis? ("はい" means yes, "いいえ" means no.)
- EstimateMonth:Estimated period (month)
- EstimateMonth_Reason:Reason of the estimated period
- Cij_Value_KLOC:Maximum productivity (KLOC/ month)
- Cij_Reason:Reason of the maximum productivity
- bij_Value:Developer's knowledge level (%)
- bij_Reason:Reason of the developer's knowledge level
- Kij_Value_MaxHour:How many day does a developer need to master?
- Kij_Reason:Reason of the above "days"
- Eij_Developer_ActName:contents of developer's knowledge
- Eij_Developer_Value:How much is developer's knowledge level? when a developer gives up. (%)
- Eij_Required_ActName:contents of the required knowledge
- Eij_Required_Value:How much is the required knowledge level? when a developer gives up. (%)
- SimulationResult:Simulation result

Virtual Project No.1;
:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:いいえ
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:いいえ
:管理者の指定言語での開発プロジェクトの参加経験:いいえ
:管理者の経験年数:10 年以上
:管理者の入力値使用可能？:はい
:x_comment:
:EstimateMonth:12.5
:EstimateMonth_Reason:・生産管理のアプリケーションを知らない・C 言語を使うのは初めて
:Cij_Value_KLOC:0.95
:Cij_Reason:・開発環境が初めて・ポインタを知らない
:bij_Value:20
:bij_Reason: COBOL と Visual Basic でのプログラム経験がある
（本人は 10 %といっているので、それを倍にした）
:Kij_Value_MaxHour:25
:Kij_Reason: ポインタと再帰が理解できなくて苦労すると考える
:Eij_Developer_ActName:基本的な変数の型と演算
:Eij_Developer_Value:20
:Eij_Required_ActName:ポインタと再帰
:Eij_Required_Value:80
:SimulationResult:11.1

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:いいえ
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:６から１０年
:管理者の入力値使用可能？:はい
:EstimateMonth:5.6
:EstimateMonth_Reason:
:Cij_Value_KLOC:2
:Cij_Reason:VB と同じぐらいと考えた
:bij_Value:10
:bij_Reason:本人が 10 としている
:Kij_Value_MaxHour:18
:Kij_Reason: 飲み込みが悪そうだから

:Eij_Developer_ActName:言語知識
:Eij_Developer_Value:80
:Eij_Required_ActName:OS に関する知識
:Eij_Required_Value:100
:SimulationResult:6.21

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:いいえ
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:6から１０年
:管理者の入力値使用可能？:はい
:x_comment:もう少し開発者の情報を細かく増やしていただければ、もっと正確に
見積もれると思います。
:EstimateMonth:7.5
:EstimateMonth_Reason:C 言語の経験者であれば５ヶ月であるが、初心者なので
１．５倍にした。
:Cij_Value_KLOC:1.5
:Cij_Reason: プログラミングと単体テストの作業は、通常 2.0KLOC/月くらいと考える
のだが、この開発者は C 言語が始めてなので、もう少し少なく見積もった。また、過去
の実績で VB 言語が 2.0KLOC/月なので、それよりも少ないと思われるから。
また、コンピュータでの開発経験が５年なので、新しい言語といってもそこそこの
生産性は示せると思われる。
:bij_Value:10
:bij_Reason:C 言語の経験はなく、新人研修で 1ヶ月程度の内容なので、本人のアン
ケートの内容のように、10%くらいしか身についていないと思われる。
:Kij_Value_MaxHour:25
:Kij_Reason:
:Eij_Developer_ActName:変数の型と四則演算と基本的な文法
:Eij_Developer_Value:20
:Eij_Required_ActName:ポインタ操作
:Eij_Required_Value:50
:SimulationResult:7.96

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:いいえ
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:いいえ
:管理者の指定言語での開発プロジェクトの参加経験:はい

:管理者の経験年数: 0
:管理者の入力値使用可能？:はい
:x_comment:勘と経験というのも難しいものですね.
:EstimateMonth:10
:EstimateMonth_Reason:
:Cij_Value_KLOC:2.5
:Cij_Reason:開発経験における平均値が COBOL で 1.5KLOC,VB で 2KLOC であるので,
最高値はそれらより大きいと考えた.
:bij_Value:10
:bij_Reason:本人が 10%と言っており，他に参考となるデータがないから.
:bij_SELECT:アンケート知識度 C
:Kij_Value_MaxHour:83
:Kij_Reason:
:Eij_Developer_ActName:変数の型と四則演算と基本的な文法
:Eij_Developer_Value:30
:Eij_Required_ActName:ポインタ操作
:Eij_Required_Value:60
:SimulationResult:8.04

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:いいえ
:管理者の指定言語での開発プロジェクトの参加経験:いいえ
:管理者の経験年数:10 年以上
:管理者の入力値使用可能？:はい
:x_comment:
C 言語の経験がないので、お役に立てなかったように思えますが、面白かったです。
あと、パラメータに影響すると思われる開発者の属性については、複数回答を
したかったのですが、やり方がわからず、選べませんでした。すみません。
(^^; C言語の平均生産性を数値として与えていただければ、私の経験からでも、
もう少し実のある回答ができたかもしれないと思います。とにかく、研究、
:EstimateMonth:8
:EstimateMonth_Reason:
:Cij_Value_KLOC:3
:Cij_Reason:
:bij_Value:10
:bij_Reason:
:Kij_Value_MaxHour:83
:Kij_Reason:

:Eij_Developer_ActName:
:Eij_Developer_Value:30
:Eij_Required_ActName:
:Eij_Required_Value:60
:SimulationResult:7.60

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:いいえ
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:３から５年
:管理者の入力値使用可能？:はい
:EstimateMonth:6.7
:EstimateMonth_Reason:10K ÷ 1.5K/人月新しい言語を習得する上でもＶＢよりＣの方が生産性落ちると思われる。ただし経験年数５年を考慮して、新人時のＣＯＢＯＬと同じ生産性とした。
:Cij_Value_KLOC:2.5
:Cij_Reason:作業の後半で言語に習熟した場合、平均の２倍近い（勘）生産性になっていると思われる。
:bij_Value:8
:bij_Reason:新人のときの研修なので、実際は本人が思っているよりもう少し低い。
:Kij_Value_MaxHour:60
:Kij_Reason:１ヶ月で１０％程度なら、あと３ヶ月ぐらい
:Eij_Developer_ActName:基本的な文法
:Eij_Developer_Value:10
:Eij_Required_ActName:ポインタ操作
:Eij_Required_Value:50
:SimulationResult:6.41

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:３から５年
:管理者の入力値使用可能？:はい
:EstimateMonth:10
:EstimateMonth_Reason:「生産管理システム」とだけでは推測できないし、詳細設計がどの程度の記述され方をしているのかも推測できません。作成する関数の流用度、

また、システム的なテスト環境も分かりませんので、１関数に対して仕様理解～
テスト完了（全体テスト工数を按分）を２．５時間と見積もりますその他、実際の
開発等では仕様変更や、部分的な検収も随時行われる為＋αします。ただ、この開発者
の経験及びＣ言語という環境下では、１Ｋ／月が妥当でしょう。
つまり、１００００÷１０００＝１０
:Cij_Value_KLOC:2
:Cij_Reason:前出の理由を参照
:bij_Value:20
:bij_Reason:ＶＢを一般化してしまう傾向があるため、ＰＣでの開発を誤認識している
:Kij_Value_MaxHour:90
:Kij_Reason:「完全」にという言葉があいまいなため、９０日というあいまいな
答えとなってしまう。
:Eij_Developer_ActName:VB と同じようなＰＣでの開発言語の１種としか考えていない
:Eij_Developer_Value:2
:Eij_Required_ActName:なぜＣ言語で作成するかの知識
:Eij_Required_Value:22
:SimulationResult:10.2

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:いいえ
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:いいえ
:管理者の経験年数:０
:管理者の入力値使用可能？:はい
:EstimateMonth:5
:EstimateMonth_Reason:初心者なので、実際のコーディング以外にプログラム言語を
学習する時間が長くなると予測したため。
:Cij_Value_KLOC:3
:Cij_Reason:
:bij_Value:60
:bij_Reason:
:Kij_Value_MaxHour:10
:Kij_Reason:
:Eij_Developer_ActName:変数の型と四則演算と基本的な文法
:Eij_Developer_Value:20
:Eij_Required_ActName:動的メモリ割り当て
:Eij_Required_Value:40
:SimulationResult:3.49

```
:-----------------------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:いいえ
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:いいえ
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:10 年以上
:管理者の入力値使用可能？:はい
:EstimateMonth:15
:EstimateMonth_Reason:
:Cij_Value_KLOC:0.95
:Cij_Reason:
:bij_Value:20
:bij_Reason:
:Kij_Value_MaxHour:200
:Kij_Reason:
:Eij_Developer_ActName:基本的な変数の型と演算
:Eij_Developer_Value:20
:Eij_Required_ActName:ポインタと再帰
:Eij_Required_Value:80
:SimulationResult:14.6


:-----------------------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:いいえ
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数: 1 から 2 年
:管理者の入力値使用可能？:はい
:EstimateMonth:5
:EstimateMonth_Reason:
:Cij_Value_KLOC:2.5
:Cij_Reason:
:bij_Value:70
:bij_Reason:
:Kij_Value_MaxHour:200
:Kij_Reason:
:Eij_Developer_ActName:
:Eij_Developer_Value:20
```

:Eij_Required_ActName:
:Eij_Required_Value:50
:SimulationResult:5.16

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:６から１０年
:管理者の入力値使用可能？:はい
:EstimateMonth:15
:EstimateMonth_Reason:3ヶ月を全く無駄な作業に使うだろう
1ヶ月はＣ言語の理解に苦しむ時間
2ヶ月をGUIも構築に使うだろう(学習込み)
　　最終的にダイアログだけで済ますのが落ち
3ヶ月をサーバ側の構築に使うだろ
:Cij_Value_KLOC:2
:Cij_Reason:
:bij_Value:30
:bij_Reason:
:Kij_Value_MaxHour:200
:Kij_Reason:
:Eij_Developer_ActName:
:Eij_Developer_Value:30
:Eij_Required_ActName:
:Eij_Required_Value:50
:SimulationResult:14.2

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:０
:管理者の入力値使用可能？:はい
:EstimateMonth:6
:EstimateMonth_Reason:過去の実績に基づいて。
:Cij_Value_KLOC:1
:Cij_Reason:

:bij_Value:10
:bij_Reason:
:Kij_Value_MaxHour:100
:Kij_Reason:
:Eij_Developer_ActName:変数の型と四則演算と基本的な文法
:Eij_Developer_Value:20
:Eij_Required_ActName:ポインタ操作
:Eij_Required_Value:50
:SimulationResult:14.9

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:10 年以上
:管理者の入力値使用可能？:はい
:EstimateMonth:4
:EstimateMonth_Reason:そこそこのプログラマーで５ＫＬ/MMと考えますが
言語知識が１ヶ月のため、２倍は掛かると読みました
:Cij_Value_KLOC:3
:Cij_Reason:
:bij_Value:10
:bij_Reason:
:Kij_Value_MaxHour:40
:Kij_Reason:
:Eij_Developer_ActName:言語仕様
:Eij_Developer_Value:10
:Eij_Required_ActName:通常関数群
:Eij_Required_Value:50
:SimulationResult:4.86

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:いいえ
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:６から１０年
:管理者の入力値使用可能？:はい
:EstimateMonth:11

:EstimateMonth_Reason:お勉強に 1ヶ月＋ 1kstep/月
:Cij_Value_KLOC:1
:Cij_Reason:
:bij_Value:70
:bij_Reason:
:Kij_Value_MaxHour:90
:Kij_Reason:
:Eij_Developer_ActName:基本的な文法
:Eij_Developer_Value:30
:Eij_Required_ActName:動的メモリ管理
:Eij_Required_Value:70
:SimulationResult:10.4

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:10 年以上
:管理者の入力値使用可能？:はい
:EstimateMonth:11
:EstimateMonth_Reason:原工数として５人月。これに、アプリケーション経験、
システム設計経験、C 言語経験の各不足分を加味。
:Cij_Value_KLOC:1.5
:Cij_Reason:
:bij_Value:10
:bij_Reason:
:Kij_Value_MaxHour:60
:Kij_Reason:
:Eij_Developer_ActName:
:Eij_Developer_Value:20
:Eij_Required_ActName:
:Eij_Required_Value:50
:SimulationResult:9.50

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい

:管理者の経験年数:３から５年
:管理者の入力値使用可能？:はい
:EstimateMonth:5
:EstimateMonth_Reason:経験的に、Cのプログラム開発の生産性が一ヶ月２０００
ステップ程度と思っているから。
:Cij_Value_KLOC:2.5
:Cij_Reason:
:bij_Value:60
:bij_Reason:
:Kij_Value_MaxHour:10
:Kij_Reason:
:Eij_Developer_ActName:cobol
:Eij_Developer_Value:30
:Eij_Required_ActName:ポインタなど
:Eij_Required_Value:50
:SimulationResult:4.17

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:３から５年
:管理者の入力値使用可能？:はい
:EstimateMonth:15.625
:EstimateMonth_Reason:640step/月で見積もり。最もよく利用するソフトハウスの
クライアント/サーバ基幹業務システムの標準生産性*0.8
:Cij_Value_KLOC:0.8
:Cij_Reason:
:bij_Value:50
:bij_Reason:
:Kij_Value_MaxHour:60
:Kij_Reason:
:Eij_Developer_ActName:ロジックを正しく組める
:Eij_Developer_Value:50
:Eij_Required_ActName:パフォーマンスチューニング
:Eij_Required_Value:90
:SimulationResult:13.3

:----------:

Virtual Project No.2
:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:６から１０年
:管理者の入力値使用可能？:はい
:EstimateMonth:10
:EstimateMonth_Reason:VisualBasic が3 KLOC/月でC が1 KLOC/月と考えた。
加えて，学習のために1ヶ月半とった。
:Cij_Value_KLOC:2
:Cij_Reason:
:bij_Value:40
:bij_Reason:
:Kij_Value_MaxHour:200
:Kij_Reason:
:Eij_Developer_ActName:基本文法
:Eij_Developer_Value:10
:Eij_Required_ActName:継承の概念
:Eij_Required_Value:40
:SimulationResult:9.29

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:いいえ
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:３から５年
:管理者の入力値使用可能？:はい
:EstimateMonth:30
:EstimateMonth_Reason:開発者の過去の経験から　ドキュメント作成に半年
VisualBasic 部分に半年　C　の習熟に半年　C　での開発に1年の合計
:Cij_Value_KLOC:1
:Cij_Reason:
:bij_Value:30
:bij_Reason:
:Kij_Value_MaxHour:700

:Kij_Reason:
:Eij_Developer_ActName:非オブジェクト思考言語での開発
:Eij_Developer_Value:30
:Eij_Required_ActName:オブジェクト思考をベースとした設計
:Eij_Required_Value:80
:SimulationResult:22.4

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:６から１０年
:管理者の入力値使用可能？:はい
:EstimateMonth:9
:EstimateMonth_Reason:VB は 2k/MM C　は 1k/MM だが学習に 1.5ヶ月かかる
:Cij_Value_KLOC:1.5
:Cij_Reason:
:bij_Value:20
:bij_Reason:
:Kij_Value_MaxHour:180
:Kij_Reason:
:Eij_Developer_ActName:基本的な文法
:Eij_Developer_Value:20
:Eij_Required_ActName:動的バインディング
:Eij_Required_Value:50
:SimulationResult:13.5

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:いいえ
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:１から２年
:管理者の入力値使用可能？:はい
:EstimateMonth:12
:EstimateMonth_Reason:C　言語の習熟：0.5 人月設計（テストコーディング含む)：
1.5 人月 VisualBasic コーディング：3 人月 C　コーディング：5 人月結合テスト：2 人月
:Cij_Value_KLOC:2.0

:Cij_Reason:
:bij_Value:50
:bij_Reason:
:Kij_Value_MaxHour:30
:Kij_Reason:
:Eij_Developer_ActName:配列以外の変数宣言と基本的な文法
:Eij_Developer_Value:10
:Eij_Required_ActName:リストを扱う
:Eij_Required_Value:20
:SimulationResult:6.36

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:いいえ
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数: 0
:管理者の入力値使用可能？:はい
:EstimateMonth:10.5
:EstimateMonth_Reason:If the number of the input character is too many,
it may be error of submitting.Please write briefly. If you want to describe
the reason, please use the comment spaceon the right side page.
:Cij_Value_KLOC:2
:Cij_Reason:
:bij_Value:40
:bij_Reason:
:Kij_Value_MaxHour:120
:Kij_Reason:
:Eij_Developer_ActName:VisualBasic structure and grammer, WindowsNT
fundamentals, and basic grammar of C
:Eij_Developer_Value:40
:Eij_Required_ActName:C grammer, structure, and control flow
:Eij_Required_Value:50
:SimulationResult:13.8

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:いいえ
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい

:管理者の経験年数:３から５年
:管理者の入力値使用可能？:はい
:EstimateMonth:10
:EstimateMonth_Reason:C習得に 1.5, VB 習得に 0.5, Cプログラム設計と開発が
3.0, VBプログラム設計と開発が3.0, テストおよびバグ修正で1.0,
リスク分として 1.0
:Cij_Value_KLOC:1.1
:Cij_Reason:
:bij_Value:20
:bij_Reason:
:Kij_Value_MaxHour:40
:Kij_Reason:
:Eij_Developer_ActName:ポインタ操作の必要なプログラミング
:Eij_Developer_Value:50
:Eij_Required_ActName:クラスの作成
:Eij_Required_Value:75
:SimulationResult:10.9

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:いいえ
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:１から２年
:管理者の入力値使用可能？:はい
:EstimateMonth:15.5
:EstimateMonth_Reason:下記の必要な知識習得時間、およびプログラミングと
ドキュメント作成の生産性、さらに技術難題解決能力に基づいた。
:Cij_Value_KLOC:1.2
:Cij_Reason:
:bij_Value:25
:bij_Reason:
:Kij_Value_MaxHour:30
:Kij_Reason:
:Eij_Developer_ActName:変数宣言・定義とプログラム構造および文法
:Eij_Developer_Value:30
:Eij_Required_ActName:クラスおよびライブラリー
:Eij_Required_Value:60
:SimulationResult:9.47

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:いいえ
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:６から１０年
:管理者の入力値使用可能？:はい
:EstimateMonth:9.1
:EstimateMonth_Reason: ＶＢ部分：２．５ヶ月、Ｃ＋＋部分：６．６ヶ月
:Cij_Value_KLOC:2
:Cij_Reason:
:bij_Value:50
:bij_Reason:
:Kij_Value_MaxHour:60
:Kij_Reason:
:Eij_Developer_ActName:言語
:Eij_Developer_Value:80
:Eij_Required_ActName:全て（ＯＳ等）
:Eij_Required_Value:100
:SimulationResult:6.55

Virtual Project No.3
:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:いいえ
:管理者の指定言語での開発プロジェクトの参加経験:いいえ
:管理者の経験年数:６から１０年
:管理者の入力値使用可能？:はい
:EstimateMonth:6.67
:EstimateMonth_Reason:1.5KLOC/月と考えた。COBOL や Mainframe に関する知識の習得はほとんど必要ないと考える。
:Cij_Value_KLOC:1.5
:Cij_Reason:
:bij_Value:90
:bij_Reason:
:Kij_Value_MaxHour:200
:Kij_Reason:
:Eij_Developer_ActName:

:Eij_Developer_Value:20
:Eij_Required_ActName:
:Eij_Required_Value:50
:SimulationResult:6.90

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:いいえ
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:３から５年
:管理者の入力値使用可能？:はい
:EstimateMonth:7.5
:EstimateMonth_Reason:　担当者の開発能力は平均１．５ｋｓ／月程度。この値で求めると　１０／１．５＝６．５ヶ月　業務／マシン／言語については、３年程度の経験者である。　性格的には問題解決力にかける為、あと工程にてトラブルを発見し、手戻り工数が発生しそう。
:Cij_Value_KLOC:1.5
:Cij_Reason:
:bij_Value:50
:bij_Reason:
:Kij_Value_MaxHour:20
:Kij_Reason:
:Eij_Developer_ActName:基本的な文法
:Eij_Developer_Value:50
:Eij_Required_ActName:性能向上
:Eij_Required_Value:70
:SimulationResult:7.25

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:いいえ
:管理者の経験年数:０
:管理者の入力値使用可能？:はい
:EstimateMonth:5.5
:EstimateMonth_Reason:工程が製造～内部結合テストまでと見てよい。
３年間で０から始めて1.8KLOC/月の生産性があったことから２年のブランク
を考慮して2.0KLOC/月になっているものと思われる。開発する物件が10KLOCなので、

単純に割り算して5ヶ月、さらに必要な未知の技術を習得するのに半月と見て5.5ヶ月と思われる。
:Cij_Value_KLOC:2.0
:Cij_Reason:
:bij_Value:40
:bij_Reason:
:Kij_Value_MaxHour:20
:Kij_Reason:
:Eij_Developer_ActName:ホスト内処理
:Eij_Developer_Value:40
:Eij_Required_ActName:ホスト間等の通信技術
:Eij_Required_Value:75
:SimulationResult:5.46

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:10年以上
:管理者の入力値使用可能？:はい
:EstimateMonth:5
:EstimateMonth_Reason:(1)COLBOLの作成経験はあるので、開発生産性は伸びると思う
(2)10k程度の設計については、当人の努力で短期間（20日）前後でできる可能性がある。
(3)COBOLのある部分についは同一となる可能性がある。
(4) 定義、定義、working-storageの定義についても、全部ので共通化が
できる。
(5) 定義についても、当人のSTEPについては　同一にできます。
結論としては、設計に十分時間を取り、各についてはCOPY文定義の中に、
別の定義分を設定したりする。の文についても、同一です。
つまり、どのようにも、なります。多分、５ｋほどのコーデイングで１５ｋとか、
２０Ｋとか、いくらでも組めます。SMGS：山内@シ生技.三菱電機の弟子より
:Cij_Value_KLOC:500
:Cij_Reason:
:bij_Value:35
:bij_Reason:
:Kij_Value_MaxHour:30
:Kij_Reason:
:Eij_Developer_ActName:COBOL言語のみ
:Eij_Developer_Value:40

:Eij_Required_ActName:DB 定義や定義
:Eij_Required_Value:20
:SimulationResult:1.17

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:いいえ
:管理者の経験年数:10 年以上
:管理者の入力値使用可能？:はい
:EstimateMonth:7.0
:EstimateMonth_Reason:　彼は 3 年間に 54Kloc のバンキングシステムのプログラムを作成している。この割合が継続すると期待できるので。(コメント欄に、さらに付記)
:Cij_Value_KLOC:1.5
:Cij_Reason:
:bij_Value:80
:bij_Reason:
:Kij_Value_MaxHour:15
:Kij_Reason:
:Eij_Developer_ActName:帳票の編集
:Eij_Developer_Value:30
:Eij_Required_ActName:一般のファイル入出力
:Eij_Required_Value:80
:SimulationResult:6.77

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数: 3 から 5 年
:管理者の入力値使用可能？:はい
:EstimateMonth:5.5
:EstimateMonth_Reason:開発期間を 5 ヶ月として、多少なりとも変更があると見越し 0．5 ヶ月加算した。
:Cij_Value_KLOC:2.0
:Cij_Reason:
:bij_Value:50
:bij_Reason:

:Kij_Value_MaxHour:25
:Kij_Reason:
:Eij_Developer_ActName:基本的な文法
:Eij_Developer_Value:50
:Eij_Required_ActName:大容量ファイル
:Eij_Required_Value:80
:SimulationResult:5.49

:----------:

:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:いいえ
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:10 年以上
:管理者の入力値使用可能？:はい
:EstimateMonth:4.5
:EstimateMonth_Reason:開発 step 10000step, 設計 0.7 人月, 開発 2.5 人月, テスト 1.0 人月, ドキュメント 0.3 人月
:Cij_Value_KLOC:3
:Cij_Reason:
:bij_Value:75
:bij_Reason:
:Kij_Value_MaxHour:50
:Kij_Reason:
:Eij_Developer_ActName:文法、命令
:Eij_Developer_Value:20
:Eij_Required_ActName:多段階添字処理
:Eij_Required_Value:80
:SimulationResult:3.5

Virtual Project No.4
:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:いいえ
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:６から１０年
:管理者の入力値使用可能？:はい

:EstimateMonth:15.5
:EstimateMonth_Reason:0.75/月で見積った。しかし，Cの学習のため2ヶ月くらいとった
:Cij_Value_KLOC:1.0
:Cij_Reason:
:bij_Value:55
:bij_Reason:
:Kij_Value_MaxHour:200
:Kij_Reason:
:Eij_Developer_ActName:
:Eij_Developer_Value:10
:Eij_Required_ActName:
:Eij_Required_Value:30
:SimulationResult:13.9

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:3から5年
:管理者の入力値使用可能？:はい
:EstimateMonth:15
:EstimateMonth_Reason:事前学習：1ヶ月，コーディング：2ヶ月，単体テスト：1ヶ月，全体テスト：1ヶ月
:Cij_Value_KLOC:4
:Cij_Reason:
:bij_Value:30
:bij_Reason:
:Kij_Value_MaxHour:800
:Kij_Reason:
:Eij_Developer_ActName:本で勉強したような基本知識
:Eij_Developer_Value:30
:Eij_Required_ActName:曖昧な設計を実コーディングに落とす
:Eij_Required_Value:80
:SimulationResult:16.5

:----------:
:NAME:

:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:10 年以上
:管理者の入力値使用可能？:はい
:EstimateMonth:24
:EstimateMonth_Reason:１ヶ月の平均生産性を５００Ｓｔｅｐと推測。この場合、２０ヶ月で完成するが、２ヶ月間をＣ言語の学習＆習熟期間、２ヶ月間をゆとり期間とした。
:Cij_Value_KLOC:1
:Cij_Reason:
:bij_Value:10
:bij_Reason:
:Kij_Value_MaxHour:1000
:Kij_Reason:
:Eij_Developer_ActName:変数の型と四則演算と基本的な文法
:Eij_Developer_Value:20
:Eij_Required_ActName:システムコール
:Eij_Required_Value:100
:SimulationResult:28.3

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:６から１０年
:管理者の入力値使用可能？:はい
:EstimateMonth:13
:EstimateMonth_Reason:プログラム構成設計＝１ｗ，
プログラム詳細設計＝３ｗ，プログラミング＝２ｗ，単体テスト＝２ｗ，
結合テスト＝１ｗ，その他　仕様確認／理解、テスト後の設計書修正等２ｗ
:Cij_Value_KLOC:3
:Cij_Reason:
:bij_Value:10
:bij_Reason:
:Kij_Value_MaxHour:365
:Kij_Reason:
:Eij_Developer_ActName:基本文法
:Eij_Developer_Value:20

:Eij_Required_ActName:マクロ（ＯＳ提供)、ＳＶＣ命令、ハード構造
:Eij_Required_Value:80
:SimulationResult:12.1

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:３から５年
:管理者の入力値使用可能？:はい
:EstimateMonth:8
:EstimateMonth_Reason:・Ｃ言語習得に１ヶ月・システムプログラムの基本理解に
１ヶ月　システムソフト開発にはかなりの専門知識が必要であり、アプリケーション
開発の経験だけでは不充分。開発に６ヶ月：(習熟者の３倍) 基本からじっくりと
取り組むタイプの人であれば、同じ経験値でも　習熟者の２倍程度と思われるが、
この場合は３倍とした。　正直なところ、この人材にはこの仕事はアサインしない。
:Cij_Value_KLOC:2
:Cij_Reason:
:bij_Value:10
:bij_Reason:
:Kij_Value_MaxHour:40
:Kij_Reason:
:Eij_Developer_ActName:文法理解
:Eij_Developer_Value:20
:Eij_Required_ActName:for(),signal() 等のシステムコールの利用
:Eij_Required_Value:80
:SimulationResult:6.02

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:いいえ
:管理者の経験年数:０
:管理者の入力値使用可能？:はい
:EstimateMonth:7.0
:EstimateMonth_Reason:８０時間の学習時間と知識量は不足するが Cobol での生産性
:Cij_Value_KLOC:1.5
:Cij_Reason:

:bij_Value:10
:bij_Reason:
:Kij_Value_MaxHour:20
:Kij_Reason:
:Eij_Developer_ActName:基本機能
:Eij_Developer_Value:20
:Eij_Required_ActName:高度な応用
:Eij_Required_Value:80
:SimulationResult:7.26

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:いいえ
:管理者の経験年数:0
:管理者の入力値使用可能?:はい
:EstimateMonth:7.0
:EstimateMonth_Reason:80時間の学習時間と知識量は不足するがCobolでの生産性
:Cij_Value_KLOC:1.5
:Cij_Reason:
:bij_Value:10
:bij_Reason:
:Kij_Value_MaxHour:20
:Kij_Reason:
:Eij_Developer_ActName:基本機能
:Eij_Developer_Value:20
:Eij_Required_ActName:高度な応用
:Eij_Required_Value:80
:SimulationResult:7.22

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:いいえ
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:1から2年
:管理者の入力値使用可能?:はい
:EstimateMonth:8
:EstimateMonth_Reason:C言語の場合、プログラム平均生産性は1.2KLOC/月と考え、

10KLOC で約８ケ月かかる。ドキュメント平均生産性は基本的に変わらないので 18
ページ/月と考え、50 ページで約３ケ月かかる。バンキングシステムの場合を考えると、
これらの作業はパラレルにこなせると予想し、８ケ月と推定した。
:Cij_Value_KLOC:1.2
:Cij_Reason:
:bij_Value:30
:bij_Reason:
:Kij_Value_MaxHour:14
:Kij_Reason:
:Eij_Developer_ActName:制御構造、配列、型
:Eij_Developer_Value:30
:Eij_Required_ActName:ポインタ、構造体
:Eij_Required_Value:50
:SimulationResult:8.91

Virtual Project No.5
:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:いいえ
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:６から１０年
:管理者の入力値使用可能？:はい
:EstimateMonth:20
:EstimateMonth_Reason:HTML 文書を 20 ページ/月，JAVA と CGI を 1 KLOC/月とし，
学習時間を 5ヶ月とした。
:Cij_Value_KLOC:2
:Cij_Reason:
:bij_Value:10
:bij_Reason:
:Kij_Value_MaxHour:360
:Kij_Reason:
:Eij_Developer_ActName:
:Eij_Developer_Value:10
:Eij_Required_ActName:
:Eij_Required_Value:40
:SimulationResult:24.7

:----------:

:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:いいえ
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:１から２年
:管理者の入力値使用可能？:はい
:EstimateMonth:9
:EstimateMonth_Reason:本人のいう通り新しいもの好きではあるようだが、
探求心および技術勘に弱いと判断。また、基礎知識も欠けているので、製品の
差し戻しがないレベルにするためには４ヶ月は直接関係のない基礎学習に、
２ヶ月を機能単体レベルの実験と設計、３ヶ月を実際の開発・デバッグ期間に
当てる必要があると推測します。
:Cij_Value_KLOC:2
:Cij_Reason:
:bij_Value:0
:bij_Reason:
:Kij_Value_MaxHour:252
:Kij_Reason:
:Eij_Developer_ActName:基本構文
:Eij_Developer_Value:20
:Eij_Required_ActName:ドキュメントオブジェクトモデルやオブジェクトの動的結合に
関する知識
:Eij_Required_Value:70
:SimulationResult:13.7

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:いいえ
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:１から２年
:管理者の入力値使用可能？:はい
:EstimateMonth:15
:EstimateMonth_Reason:製造工程　C　1ヶ月、JAVA　2ヶ月　HTML　0.5ヶ月の
計3.5ヶ月これと同期間の設計工程＋これの２倍のテスト工程を加え、14ヶ月
ここに言語習得期間１ヶ月を加え計１５ヶ月
:Cij_Value_KLOC:0.75
:Cij_Reason:
:bij_Value:20
:bij_Reason:

:Kij_Value_MaxHour:30
:Kij_Reason:
:Eij_Developer_ActName:基本的な文法
:Eij_Developer_Value:20
:Eij_Required_ActName:例外処理
:Eij_Required_Value:70
:SimulationResult:14.2

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:10 年以上
:管理者の入力値使用可能？:はい
:EstimateMonth:12.8
:EstimateMonth_Reason:
:Cij_Value_KLOC:1.8
:Cij_Reason:
:bij_Value:5
:bij_Reason:
:Kij_Value_MaxHour:180
:Kij_Reason:
:Eij_Developer_ActName:
:Eij_Developer_Value:20
:Eij_Required_ActName:
:Eij_Required_Value:60
:SimulationResult:13.0

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:６から１０年
:管理者の入力値使用可能？:はい
:EstimateMonth:12.8
:EstimateMonth_Reason
:Cij_Value_KLOC:2.0
:Cij_Reason:

:bij_Value:0
:bij_Reason:
:Kij_Value_MaxHour:200
:Kij_Reason:
:Eij_Developer_ActName:
:Eij_Developer_Value:30
:Eij_Required_ActName:
:Eij_Required_Value:60
:SimulationResult:16.9

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:いいえ
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:６から１０年
:管理者の入力値使用可能？:はい
:EstimateMonth:12
:EstimateMonth_Reason:月平均 1kstep だが C 言語に 1ヶ月そのあとに Java が 1ヶ月弱 HTML は 1 週間もかからないとしました。
:Cij_Value_KLOC:1
:Cij_Reason:
:bij_Value:0
:bij_Reason:
:Kij_Value_MaxHour:180
:Kij_Reason:
:Eij_Developer_ActName:基本的な文法
:Eij_Developer_Value:20
:Eij_Required_ActName:クラス構造の変更
:Eij_Required_Value:80
:SimulationResult:15.4

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:０
:管理者の入力値使用可能？:はい
:EstimateMonth:7

:EstimateMonth_Reason:JAVA,cgi の開発が 1.5K/月で、6ヶ月、諸々で 1 月の計 7ヶ月
:Cij_Value_KLOC:1.5
:Cij_Reason:
:bij_Value:50
:bij_Reason:
:Kij_Value_MaxHour:20
:Kij_Reason:
:Eij_Developer_ActName:シンタックス
:Eij_Developer_Value:20
:Eij_Required_ActName:手続き、構造化
:Eij_Required_Value:50
:SimulationResult:7.07

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数: 1 から 2 年
:管理者の入力値使用可能？:はい
:EstimateMonth:8
:EstimateMonth_Reason:他者が全く支援しない状況を仮定する.
これまでの経験がほとんど活かせないので，かなり時間がかかる.
:Cij_Value_KLOC:2.5
:Cij_Reason:
:bij_Value:15
:bij_Reason:
:Kij_Value_MaxHour:60
:Kij_Reason:
:Eij_Developer_ActName:表層的な文法知識と基本プログラミング
:Eij_Developer_Value:30
:Eij_Required_ActName:履歴依存性の強い部分の考察
:Eij_Required_Value:70
:SimulationResult:6.11

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい

:管理者の経験年数:１から２年
:管理者の入力値使用可能？:はい
:EstimateMonth:8
:EstimateMonth_Reason:簡易版の項番１０の理由を見てください.
:Cij_Value_KLOC:2
:Cij_Reason:
:bij_Value:20
:bij_Reason:
:Kij_Value_MaxHour:60
:Kij_Reason:
:Eij_Developer_ActName:文法＆基本的プログラミング作法
:Eij_Developer_Value:20
:Eij_Required_ActName:履歴依存性の高い部分の検討
:Eij_Required_Value:70
:SimulationResult:6.46

:----------:
:NAME:
:Adress:k-ishida@atomsystem.co.jp
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:６から１０年
:管理者の入力値使用可能？:はい
:EstimateMonth:10.7
:EstimateMonth_Reason:
:Cij_Value_KLOC:0.952
:Cij_Reason:
:bij_Value:30
:bij_Reason:
:Kij_Value_MaxHour:30
:Kij_Reason:
:Eij_Developer_ActName:当該開発環境操作とＣ言語
:Eij_Developer_Value:30
:Eij_Required_ActName:プログラム文法把握
:Eij_Required_Value:70
:SimulationResult:11.3

:----------:
:NAME:

:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:3から5年
:管理者の入力値使用可能?:はい
:EstimateMonth:12
:EstimateMonth_Reason:HTMLによる開発が2人月、JAVAによる開発が5人月、
cgi開発が3人月、テスト2人月
:Cij_Value_KLOC:1
:Cij_Reason:
:bij_Value:20
:bij_Reason:
:Kij_Value_MaxHour:40
:Kij_Reason:
:Eij_Developer_ActName:データ型
:Eij_Developer_Value:20
:Eij_Required_ActName:マルチスレッド処理
:Eij_Required_Value:60
:SimulationResult:11.3

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:10年以上
:管理者の入力値使用可能?:はい
:EstimateMonth:7.0
:EstimateMonth_Reason:全体ステップ数:10000step, 画面設計:0.5人月,
PG HTML:0.5人月, アプレット:2.5人月, CGI:1.0 人月, テスト 1.0 人月,
ドキュメント 0.5人月
:Cij_Value_KLOC:1.5
:Cij_Reason:
:bij_Value:20
:bij_Reason:
:Kij_Value_MaxHour:200
:Kij_Reason:
:Eij_Developer_ActName:変数の型と四則演算と基本的な文法
:Eij_Developer_Value:20
:Eij_Required_ActName:配列、ポインタ操作

:Eij_Required_Value:70
:SimulationResult:11.2

Virtual Project No.6

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:いいえ
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:いいえ
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:６から１０年
:管理者の入力値使用可能？:はい
:EstimateMonth:24
:EstimateMonth_Reason:
:Cij_Value_KLOC:4.0
:Cij_Reason:
:bij_Value:0
:bij_Reason:
:Kij_Value_MaxHour:360
:Kij_Reason:
:Eij_Developer_ActName:
:Eij_Developer_Value:10
:Eij_Required_ActName:
:Eij_Required_Value:35
:SimulationResult:28.7

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:６から１０年
:管理者の入力値使用可能？:はい
:EstimateMonth:15
:EstimateMonth_Reason:開発者は執着性は弱いものの要領は良さそうだという印象を受ける。一方、アセンブラプログラムは高級言語を使っての開発より、粘り強さが求められることが多い。開発者は、MainFrame 経験ありとはいえ VM の環境設定までやっていたとはみえない（アセンブラ感覚はない??）。また、アセンブラは元来プログラム効率が悪い。ということで、前半ふてく

されて効率あがらず後半要領のよさを発揮、でだいたい 15 月。
:Cij_Value_KLOC:2
:Cij_Reason:
:bij_Value:0
:bij_Reason:
:Kij_Value_MaxHour:250
:Kij_Reason:
:Eij_Developer_ActName:コーディング規則
:Eij_Developer_Value:5
:Eij_Required_ActName:ハード割り込み制御
:Eij_Required_Value:70
:SimulationResult:11.8

:----------:

:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:いいえ
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数: 0
:管理者の入力値使用可能？:いいえ
:EstimateMonth:14
:EstimateMonth_Reason:理由はあまり無いのですが、勘。です。
:Cij_Value_KLOC:0.5
:Cij_Reason:
:bij_Value:0
:bij_Reason:
:Kij_Value_MaxHour:30
:Kij_Reason:
:Eij_Developer_ActName:文法
:Eij_Developer_Value:30
:Eij_Required_ActName:割り込み処理のテクニックやマルチタスク処理の経験
:Eij_Required_Value:70
:SimulationResult:21.5

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:いいえ

:管理者の経験年数: 6 から 1 0 年
:管理者の入力値使用可能？:はい
:EstimateMonth:20.5
:EstimateMonth_Reason:基本的な学習項目は 0.5ヶ月で可能だとおもうが、
経験がないとコーディング・デバッグに時間がかかるので 1ヶ月 500 ステップ
ぐらいでしょう
:Cij_Value_KLOC:0.5
:Cij_Reason:
:bij_Value:0
:bij_Reason:
:Kij_Value_MaxHour:30
:Kij_Reason:
:Eij_Developer_ActName:文法
:Eij_Developer_Value:30
:Eij_Required_ActName:例外のデバッグ
:Eij_Required_Value:85
:SimulationResult:21.1

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:いいえ
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数: 3 から 5 年
:管理者の入力値使用可能？:はい
:EstimateMonth:29
:EstimateMonth_Reason:言語学習に 1ヶ月、構成設計に 2ヶ月、コーディングに
２０ヶ月、単体、結合テストに 6ヶ月と見込む。アセンブラはテストで問題
解決に当たらなければ習熟効果は低いため、テスト期間は長くかかる。
:Cij_Value_KLOC:0.5
:Cij_Reason:
:bij_Value:40
:bij_Reason:
:Kij_Value_MaxHour:365
:Kij_Reason:
:Eij_Developer_ActName:命令セットの理解
:Eij_Developer_Value:10
:Eij_Required_ActName:スタック操作
:Eij_Required_Value:50
:SimulationResult:26.3

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:いいえ
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:0
:管理者の入力値使用可能？:はい
:EstimateMonth:20
:EstimateMonth_Reason:上流から下流にかけての全設計における生産性は
400 ～ 500step/月と想定する。このプロジェクト規模は 10000step なので、
この生産性だと 20 ～ 25ヶ月となる。但し作業は一人で行うので、コミュニケー
ションロスはなく、一応 500step/月の生産性を使用した。
（非常に大雑把な見積りです、悪しからず。）
:Cij_Value_KLOC:0.5
:Cij_Reason:
:bij_Value:0
:bij_Reason:
:Kij_Value_MaxHour:30
:Kij_Reason:
:Eij_Developer_ActName:マニュアルを読めば書ける
:Eij_Developer_Value:50
:Eij_Required_ActName:メモリ、速度の最適化
:Eij_Required_Value:70
:SimulationResult:23.0

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:いいえ
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:0
:管理者の入力値使用可能？:はい
:EstimateMonth:20
:EstimateMonth_Reason:)
:Cij_Value_KLOC:0.5
:Cij_Reason:
:bij_Value:0
:bij_Reason:
:Kij_Value_MaxHour:30

:Kij_Reason:
:Eij_Developer_ActName:マニュアルを読めば書ける
:Eij_Developer_Value:50
:Eij_Required_ActName:メモリ、速度の最適化
:Eij_Required_Value:70
:SimulationResult:23.1

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:いいえ
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:10 年以上
:管理者の入力値使用可能？:はい
:EstimateMonth:15
:EstimateMonth_Reason:全工程の開発生産性を４００ｓｔｅｐ／人月と見積もり、
構造設計から結合テストまでを全体の６０％と見積もった。
:Cij_Value_KLOC:0.8
:Cij_Reason:
:bij_Value:10
:bij_Reason:
:Kij_Value_MaxHour:30
:Kij_Reason:
:Eij_Developer_ActName:言葉の知識
:Eij_Developer_Value:0
:Eij_Required_ActName:ハードウェア知識
:Eij_Required_Value:50
:SimulationResult:13.4

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数: 0
:管理者の入力値使用可能？:はい
:EstimateMonth:36
:EstimateMonth_Reason:未経験者がアセンブラを理解するのは意外に時間を要する。
チップが変われば言語仕様が全くことなるため、習熟者にとっても新規チップ
（ましてやカスタムチップになるとさらに）のプログラム開発は困難である。

サンプルプログラムや習熟者からの教育の有無によっても大きく異なると思われる。
実際に開発して一番の問題となるのはテストに要する時間であるが、テスト用に別の
高級言語でシミュレータを作成する必要もあり、アセンブラ意外の言語理解がカギを
握る場合もある。
:Cij_Value_KLOC:1
:Cij_Reason:
:bij_Value:30
:bij_Reason:
:Kij_Value_MaxHour:30
:Kij_Reason:
:Eij_Developer_ActName:基本的な文法とレジスタ演算
:Eij_Developer_Value:20
:Eij_Required_ActName:複数条件分岐
:Eij_Required_Value:70
:SimulationResult:10.6

:----------:
:NAME:
:バーチャルプロジェクトのソフトウエアに近いプロジェクトの参加経験:はい
:バーチャルプロジェクトにの動作環境に近いプロジェクトの参加経験:はい
:管理者の指定言語での開発プロジェクトの参加経験:はい
:管理者の経験年数:10 年以上
:管理者の入力値使用可能？:はい
:EstimateMonth:10
:EstimateMonth_Reason:潜在的能力の高い初心者が単独でアセンブラでソフト開発を
行うと仮定して言語だけでなく組込まれるハード仕様、開発支援環境の理解も含めて、
経験から上手く行って平均生産性を１ＫＬとして計算した。
:Cij_Value_KLOC:1
:Cij_Reason:
:bij_Value:0
:bij_Reason:
:Kij_Value_MaxHour:40
:Kij_Reason:
:Eij_Developer_ActName:四則演算、基本文法
:Eij_Developer_Value:50
:Eij_Required_ActName:割り込み制御
:Eij_Required_Value:80
:SimulationResult:12.4