

A Petri-Net Based Modeling Approach to Concurrent Software Engineering Tasks*

JAU-JI SHEN, S. WESLEY CHANGCHIEH⁺ AND TAO-YUAN LIN⁺⁺

*Department of Information Management
National Formosa University
Yunlin, 632 Taiwan*

*⁺Institute of Electronic Commerce
National Chung Hsing University
Taichung, 402 Taiwan*

*⁺⁺Department of Information Management
Chaoyang University of Technology
Taichung, 413 Taiwan*

The rapid growth in demand on information systems and the uncertainties associated with the development process for such systems has revealed problems such as inappropriate human resource management, poor quality, and serious delay. Traditional project management approaches emphasize the importance of scheduling, while careful allocation of resources across projects is critical and has a direct influence on the project quality and time management. According to the report on Concurrent Engineering (CE) implementation, it has significantly improved the time to market, quality, costs, etc. in the manufacturing industry. This research attempts to introduce the philosophy of CE into software project management and proposes a Concurrent Project Modeling Method (CPM-M) to realize concurrent software engineering tasks. CPM-M consists of two components: Concurrent Information Sharing Framework (CISF) and Concurrent Project Modeling Diagram (CPM-D). The CISF is a conceptual framework that facilitates concurrent cross-functional sharing of information among product, activity and resource pools, while the CPM-D inherits capability from the Petri Net and further expands so as to guide the project management by simultaneously considering the three pools at each decision point of the development process. The proposed approach was illustrated and simulated on the development of a web-based campaign system for the marketing division of a bank.

Keywords: software development, Petri net, project management, concurrent engineering, PERT

1. INTRODUCTION

The production of large, reliable software systems remains a thicket of problems including managing a large workforce, developing engineering and management processes, planning, budgeting and scheduling the project, identifying and resolving resource conflicts, monitoring the entire project continuously and applying corrective actions as new

Received February 8, 2002; revised March 13, 2003; accepted May 31, 2004.

Communicated by Michael R. Lyu.

* This research was supported in part by the National Science Council of Taiwan, R.O.C., under contract No. NSC 88-2213-E-324-003.

conflicts arise [3]. Traditional project management tools provide excellent recording, reporting, and scheduling functions, such as Gantt charts, CPM (Critical Path Method), and PERT (Program Evaluation and Review Technique), but they fail to provide higher-order capabilities. Even with the advancement of research on software engineering metrics, quality, and project management, some of the important practical issues have not been taken into account or satisfactorily resolved, such as the skills levels in human resource allocation [14], the dependency among resources, activities and products, and project management tools for dynamic concurrent software project management.

According to the report on successful applications of Concurrent Engineering (CE), IBM, AT&T and Hotpoint significantly improved the time to market, quality and costs in the manufacturing industry [18]. In contrast to sequential development, CE originated from a parallel effort which simultaneously considers tasks assigned to isolated functional units and resolves conflicts early in the design stage.

This research addresses three issues at once to solve the problems in software development: software project management, concurrent engineering, and software project diagramming and modeling. A Concurrent Project Modeling Method (CPM-M) which consists of two components, Concurrent Information Sharing Framework (CISF) and CPM Diagram (CPM-D), is proposed for software development. CISF, a conceptual framework, is designed to promote CE by concurrently considering three information pools, product, activity and resource pools, and thus encourage information sharing and integration across functions. The other component CPM-D is a modeling approach to facilitate CISF. CPM-D inherits and extends the Petri-Net capability of static and dynamic modeling of a system so as to dynamically and continuously monitor the project by deliberating resource, activity and product flows simultaneously. In short, the static characteristic of the CPM-M supports the manager in planning the project and the dynamic characteristic of the CPM-M supports making effective and responsive decisions.

The remainder of this paper is organized as follows. In the next section, the related literature is discussed. In section 3, the proposed approach and an illustrative example are presented. In section 4, a real case study is introduced and some scenarios are discussed to demonstrate our proposed approach. In the last section, discussions and conclusions are provided.

2. ISSUES OF RELATED LITERATURE

2.1 Petri Net

Petri nets are graphical and mathematical modeling tools that can be used to model static and dynamic systems. Systems that can be characterized to be concurrent, asynchronous, distributed, parallel, non-deterministic and stochastic, can be effectively modeled and analyzed using Petri nets [13]. Sawhney [19] proposed five categories of Petri nets. In its original form, transition firing in a Petri net is assumed to be instantaneous. However, time can be incorporated into a Petri net to represent a delay after a transition is enabled. Such a Petri net is known as a Timed Petri Net (TPN). If the transition times are deterministic, it is called a Deterministic Timed Petri Net (DTPN). If the transition times are allowed to be generated by random variables, it is called a Stochastic Timed

Petri Net (STPN). A Petri net that contains immediate transitions, deterministic transitions and stochastic transition is called a Generalized Stochastic Petri Net (GSPN).

To illustrate a Petri net, we use a simple example. Figs. 1 (a) and (b) show the process of table fabrication. The Petri net consists of four elements: places, transitions, arcs and tokens. A place is denoted by a circle, which represents a condition such as input data, input signal, resource, condition, or buffer [13]. A transition, denoted by a bar, represents an event such as a computation step, task or transition in a Petri net [13]. Arcs are utilized to connect a place to a transition or from a transition to a place. Arcs in a Petri net can have multiplicity, which can be interpreted as a set of “ n ” parallel arcs. The fourth element called token is denoted by a solid circle. Tokens are initialized at a place and a place may contain zero or more tokens. The concept of “transition firing” allows a Petri net to simulate the dynamic behavior of a system. The transition in a Petri net can fire only when each input place has “ n ” or more specified tokens available, a condition called “enabled.” If a transition is enabled, it will digest the number of tokens from input places and generate a specified number of tokens to the output places. The number of tokens is determined by “ n ” on the arc. For example, a table to be fabricated must have a table top and four feet as described in Fig. 1 (a). There are two places, feet of the table and table top, containing four and two tokens, respectively. The condition of “fabricate transition” will not be enabled unless four table feet and one table top are obtained. After the transition fires, the table will be built in the table place, and four table feet and one table top will be consumed (Fig. 1 (b)).

Since Petri nets can be used to model static and dynamic systems, this research extends Petri nets and develops the CPM-D to better support the manager in static planning of the project and dynamic adjustment with responsive and corrective feedback.

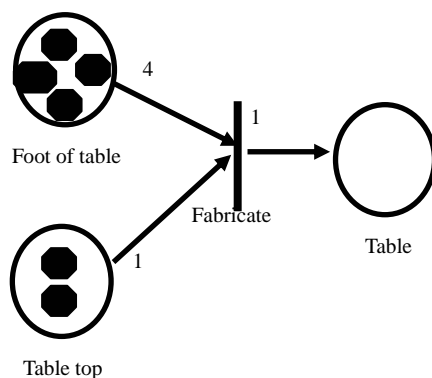


Fig. 1. (a) The Petri net model of “before a table is fabricated.”

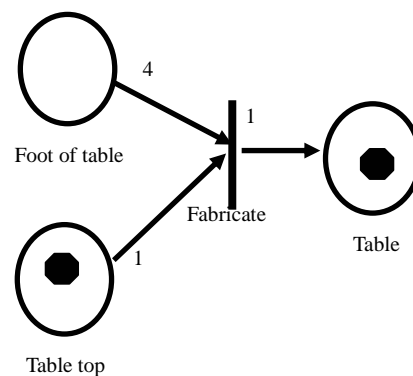


Fig. 1. (b) The Petri net model of “after a table is fabricated.”

2.2 Concurrent Engineering

Traditionally the activities in an engineering product development life cycle are usually implemented sequentially and independently. This consequently results in excessive time and cost of product development due to the isolated activities. Noore [15] men-

tioned three drawbacks of the conventional sequential approaches including no common vision, insular efforts, and late discovery of decision consequences.

A good number of concurrent engineering approaches [12, 16, 18] have been proposed. "CE is a systematic approach to the integrated, concurrent design of products and their related processes, including manufacture and support. This approach is intended from the outset to cause the developers to consider all elements of the product life cycle from conception through disposal, including quality, cost, schedule and user requirements," defined by Winner [22].

Swink [21] recommends two aspects of implementing CE, which are improving cross-functional integration and improving design analysis and decision making. In order to address all the decisions involved in a new product development process more effectively, personnel from different organizational functions must collaborate concurrently, share information, and resolve conflicts efficiently and effectively. To realize cross-functional integration, related developed methods or tools can be put into three categories. Firstly, methods like customer survey, Quality Function Deployment (QFD), and product benchmarking are used in setting and analyzing goals of an organization. Secondly, organization rules, pre-project training, project contract are used in directing and controlling cross functional integration. Last, meeting and electronic communications (e.g., e-mail) are used in encouraging communication and awareness in the organizations.

In the software industry, CE of global software products focuses on the integration of three development phases: base-product engineering, internationalization, and localization. Concurrent internationalization incorporates the capability to handle other languages and national and cultural differences into the base version; a one-time, up-front investment that obviates the need for recording and thus eliminates unnecessary and costly redesign activities, said Rafii and Perkins [17]. However, the costs and risks inherent in concurrent development are magnified when project-team members are dispersed around the globe. Aoyama [1] developed a model of distributed concurrent development which consists of three layers: collaboration layer, information layer, and operation layer. Based on his practice of the distributed concurrent development model, the major benefits include shortening the development cycle-time by 75 percent, higher stability of workload, higher utilization of workload, higher flexibility to the change of development plan, and higher quality.

2.3 Software Project Management

For project management, traditional Gantt models and activity networks like CPM and PERT are well-known methods for planning and control of a project [6-10]. The Gantt model represents activities in lines on a calendar-oriented chart. With special marks, the Gantt model can indicate project milestones, slack time, activities, and activities in the critical path. Activity networks, according to different kinds of problem, can represent projects in different forms. For deterministic problems, CPM can solve problems whose activity durations and costs are constants [4]. For probabilistic problems, PERT can be applied to problems whose activity durations and costs follow some distribution functions [4]. The major difference between PERT and CPM is that the activity of the first is indicated on the arrow, and the activity of the second is on the node [4, 23]. However, the CPM method will fail if we impose various resource restrictions on the

project network. In addition, the representation of artifacts such as documentation and codes are implicit in these schemes [3]. The current tools also treat project scheduling and resource allocation as two separate problems, although they are highly correlated.

Besides the above traditional project management methods, a number of methods have been proposed for project planning, some of which are for project planning and scheduling with limited resources such as ROT, ACTIM, ACTRES, TMROS, TG2 [5]. All of the above traditional approaches focus mainly on time scheduling, costing, and that are merely concerned with the achievement of activities of projects.

The Petri net has been used for project management [2, 3, 11]. Kumar and Ganesh [2] used a Petri net to facilitate resource allocation in project management for some commonly encountered conditions. Liu and Horwitz [11] have proposed a DesignNet model by combining a Petri net and AND-OR Graph to describe a project work breakdown structure and the specification of relationships among different types of project information (activity, product, resource, and status report information).

In concurrent engineering, the project manager is concerned not only with the scheduling of activities but also with resource and product information. From this point of view, the Petri net surpasses traditional project management methods. In Liu and Horwitz's approach [11], the product, resource and status report information are merely used to represent the constraints of start or end of an activity. In other words, it focuses on formal modeling of the activity flow and related constraints. Kumar and Ganesh's approach [2] focuses on resource allocation. Each of those two approaches individually cannot empower the manager to concurrently consider activities and integrate resource and product conditions in the organization to achieve information sharing and to facilitate decision making early in the product cycle.

Another new and exciting alternative, Critical chain, is based on the obvious fact that the longest time required for project completion can be determined by a resource that, due to limited capacity, has to perform different activities sequentially [20]. The critical chain, unlike the traditional approaches, takes resource limitations into account and is composed of sections that are dependent on precedence relationships, and other sections dependent on resource availability. Similarly, although the critical chain considers both activity precedence and resource capacity, product data are not considered in managing and optimizing the project schedule and resource allocation.

In this paper, we attempt to realize the philosophy of concurrent engineering in software development. To enhance cross-functional integration, cross-functional information sharing and collaboration is important in planning the project. Therefore, this research proposes the CPM-M. With product information, limited resource capacity and activities considered concurrently, CPM-M can support the project manager in analyzing project decision making problems like feasibility analysis, scheduling, product quality management and resource assignment.

3. CONCURRENT PROJECT MODELING METHOD (CPM-M)

The proposed Concurrent Project Modeling Method (CPM-M) consists of a Concurrent Information Sharing Framework (CISF) and a Concurrent Project Modeling Diagram (CPM-D). CISF is a conceptual framework that facilitates concurrent cross-func-

tional sharing of information among product, activity and resource pools, while the CPM-D inherits from the Petri net capability and further expands so as to guide the project management by simultaneously considering the three information pools at each decision point of the development process.

3.1 A Proposed Concurrent Information Sharing Framework

To improve cross-functional integration, all the cross-functional information in the software development process is first examined. We classify the entities of the software development process into three categories: *Activity*, *Resource* and *Product*. As most of the prior researches primarily focused on activities and/or resources, this research proposes a Concurrent Information Sharing Framework (CISF) (Fig. 2) by integrating cross-functional activities through information sharing among product, activity and resource information pools.

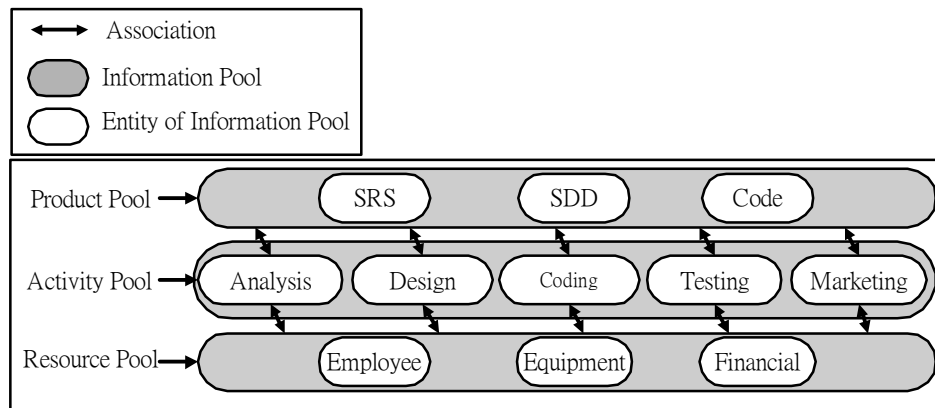


Fig. 2. A concurrent information sharing framework (CISF).

Activity is the basic unit composing a project. To implement an activity, it is important to assign the appropriate resources, such as human resources, equipment and financial resources, to the activity. Products include *finished products*, *semi-developed products* and *reference products* that result from prior activities. In the software product development life cycle, products developed in prior activities may be referenced, modified or combined with products by later activities.

The CISF separates the entities in the software project development process into three information pools: Product Information Pool, Activity Information Pool and Resource Information Pool. Therefore for CE, not only is the cross-activity information concurrently shared, but product information and resource information are also shared dynamically to reduce the project development time span and to improve product quality.

The characteristics of the three information pools are discussed below.

The Relationship of Activities

In the Activity information pool, in general the relationship between two activities can be independent or dependent. There are four basic types of dependency between activities [6]: *Finish-Start*, *Finish-Finish*, *Start-Finish* and *Start-Start*. Firstly, the Finish-Start (F-S) dependency indicates that the predecessor must finish before the successor can start. For example, the coding (predecessor) must finish before testing (successor) can start. In other words, testing has a finish-start dependency on coding. Secondly, with Finish-Finish (F-F) dependency, the predecessor must finish before the successor finishes. For example, systems documentation (successor) must reflect all changes made to program coding (predecessor). Although systems documentation can start at any time while coding is in progress, it cannot finish until coding finishes. In this case, systems documentation has a finish-finish dependency upon program coding. Thirdly, with a Start-Start (S-S) dependency, the predecessor must start before the successor can start. In this type of dependency, the successor and predecessor usually overlap. For example, information gathering (predecessor) must start before data modeling (successor) can start. In other words, data modeling has a start-start dependency upon information gathering. Lastly, with a Start-Finish (S-F) dependency, the predecessor must start before the successor can finish. For example, the purchase of a software requires that final acceptance/payment (successor) will follow an evaluation period (predecessor). But the evaluation must start before the acceptance/payment can finish.

Two Types of Products

In the product information pool, besides Finished Product according to the characteristics this research divides the products into two categories, which are *Reference Product* and *Semi-Developed Product*. Reference Products provide reference information for activities. For example, a system requirements and specifications document is a reference product which results from analysis activities and will be referenced by later activities. A Semi-Developed Product is a product generated in a prior development activity and will be continuously used in later development activities.

Two Types of Resources

In the resource information pool, the resources are classified into two categories, *Consumable Resource* and *Reusable Resource*. Consumable Resource means that the resource will be consumed during the activity, while the Reusable Resource will not. For example, financial capital is a consumable resource, and an employee is a reusable resource.

3.2 Concurrent Project Modeling Diagram (CPM-D)

Based on the CISF, a Concurrent Project Modeling Diagram (CPM-D) is developed to statically and dynamically model a software development project. CPM-D consists of six components: product place, activity, resource place, arc, transition place and token. Tokens can be divided into four categories: *activity token*, *product token*, *transition token* and *resource token*. Component notations are shown in Fig. 3 and the corresponding component attributes are listed in Table 1.

Table 1. Component attributes for CPM-D.

Component	Attribute	Format
Activity-Start transition attribute	ID	
	Requirement	{(place ID, count, constraint), (,), (,), ...}
	Output	{(place ID, count), (,), ...}
	Generation times formula of output token	(formula)
	Activity duration formula of activity token	(formula)
	Task complexity of transition token	{(place ID, task complexity), (,), (,), ...}
Activity-Working place attribute	ID	
Activity token attribute	ID	
	Time of generation	
	Activity duration	
	Semi-product tokens held	{(ID's), (,), (,), ...}
	Reference product tokens held	{(ID's), (,), (,), ...}
	Consumable resource tokens held	{(ID's), (,), (,), ...}
	Reuse resource tokens held	{(ID's), (,), (,), ...}
	Transition tokens held	{(ID's), (,), (,), ...}
Activity-End transition attribute	ID	
	Requirement	{place ID, count, constraint}
	Output	{place ID, count}
	Generation times formula of output token	(formula)
	Product quality formula of output token	{(place ID, Product quality formula), (,), (,), ...}
	Task complexity of transition token	{(place ID, task complexity), (,), (,), ...}
Product place attribute	ID	
	Type	(reference/semi-product)
	Specification	{style}
	Allocation strategy or algorithm	
	Priority of Place Output Arc	{(Arc ID, Priority), (,), ...}
Resource place attribute	ID	
	Type	(consumable/reusable)
	Specification	{style}
	Allocation strategy or algorithm	
	Priority of Place Output Arc	{(Arc ID, Priority), (,), ...}
Product token attribute	ID	
	Type	(reference/semi-product)
	Quality level	{Quality index}
	Time of generation	
Resource token attribute	ID	
	Type	(consumable/reusable)
	Capability level	{Capability index}
	Time of generation	
Transition token attribute	ID	
	Time of generation	
	Task complexity level	
Transition place attribute	ID	
	Priority of Place Output Arc	{(Arc ID, Priority), (,), ...}

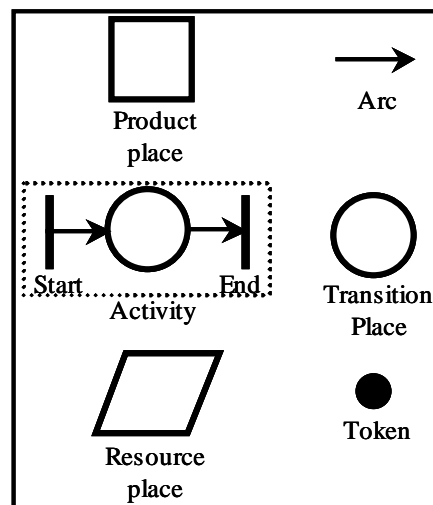


Fig. 3. Notations for CPM-D.

The definition of Arc is the same as Petri nets. Arc is utilized to connect a place to a transition or a transition to a place and can have multiplicity. Activity is a basic component in the activity information pool and consists of five elements: Start Transition, End Transition, Working Place, and two connection arcs. Once the required arcs of the Start Transition are attained, it will create an “Activity Token” at the activity Working Place until the End Transition is enabled.

Once a start transition is enabled, it will perform the following tasks:

1. Determine the generation times of output tokens, which are equal to the maximum of all the input tokens' generation times.
If the output token is an activity token, there are two tasks that must be performed. One is to determine the activity duration. To determine the duration, the proposed approach concurrently considers three factors that include resource capability level (L_r), an input task complexity level (L_t), and the product quality level (L_p). Each company should define its own empirical duration function $D = F(L_r, L_t, L_p)$ accordingly, where D is the activity duration. The other task is to calculate input product and resource token indices, which denote the input products and resources used, so as to generate an output activity token.
2. If the output token is a transition token, determine the required task complexity level. The task complexity level is predetermined in advance as shown in Table 1.
3. Fire input transition tokens and generate output tokens.

Perform the following tasks if an end transition is enabled:

1. Determine the generation times of output tokens, which can be set equal to: $\text{Max}\{(\text{activity duration of activity token entering from working place to the end transition} + \text{generation time of activity token entering from working place to the end transition}), (\text{maximum of all the other generation times of tokens to the end transition})\}$.

2. If the output token is a transition token, determine the required task complexity level. If the output token is a product (produced from this activity), determine the quality level of the output product tokens. As in the determination of the activity duration, the CPM-M can support the user to concurrently consider the three factors which affect the output product quality. Each individual company should define its own empirical quality function $Q = F(L_r, L_t, L_p)$, where Q is the quality of output product.
3. As in the start transition, if the output token is a transition token, determine the required task complexity level.
4. Fire input tokens, then generate output tokens and return the reference products and reusable resources to output places.

To define “Activity” we must identify four kinds of components as listed in Table 1: Activity start transition, Activity working place, Activity token and Activity end transition. Start transition attributes define the requirements and output of this transition. The requirements may include source place id, required number of tokens, and the constraints of requirement tokens. The constraint of a token means the attribute constraint of an input token. For example, the employee capability level may need to be greater than level B. The output attributes include destination place id and number of output tokens. Also, three formulas are needed to compute each of the generation times of output tokens, activity durations and task complexities, like start transition tasks 1 to 3 mentioned earlier. In the activity working place, we only need to determine the id. For the activity token, we need to identify the token id and other attribute values that are determined by start transition. For example, activity duration is determined by start transition. In the end transition, the only the difference compared to start transitions is that the manager must determine a formula for computing product quality as described earlier.

The dependence relationship between two activities can be labeled by transition places (shown in Fig. 4). In the transition place we must define the id and priority of the output arc in the attribute table (Table 1). Priority of an output arc indicates the priority to be compared when multiple activities require a token from this transition place. Three attributes must be described for the transition token: id, generation time and task complexity level (Table 1).

In the product information pool, the product place collects product tokens by the product type. There are some attributes that we must define in the attribute table (Table 1): product type, product specification, product allocation strategy or algorithm, and priorities of output arcs. The reference product will be returned back to the product place when the activity is finished, but the semi-developed product will not be returned.

Similar to the product place, the resource place in the resource pool defines the resource type, resource specification, resource allocation strategy or algorithm, and priorities of output arcs in the attribute table (Table 1). The reusable resource will be returned when the activity is finished, but the consumable resource will not.

Between product pool and activity pool, product tokens are transmitted. Four attributes must be defined in the attribute table (Table 1): id, product type, product quality level and generation time.

Between resource pool and activity pool, resource tokens are transmitted. Four attributes must be defined in the attribute table (Table 1): id, resource type, resource capability level and generation time.

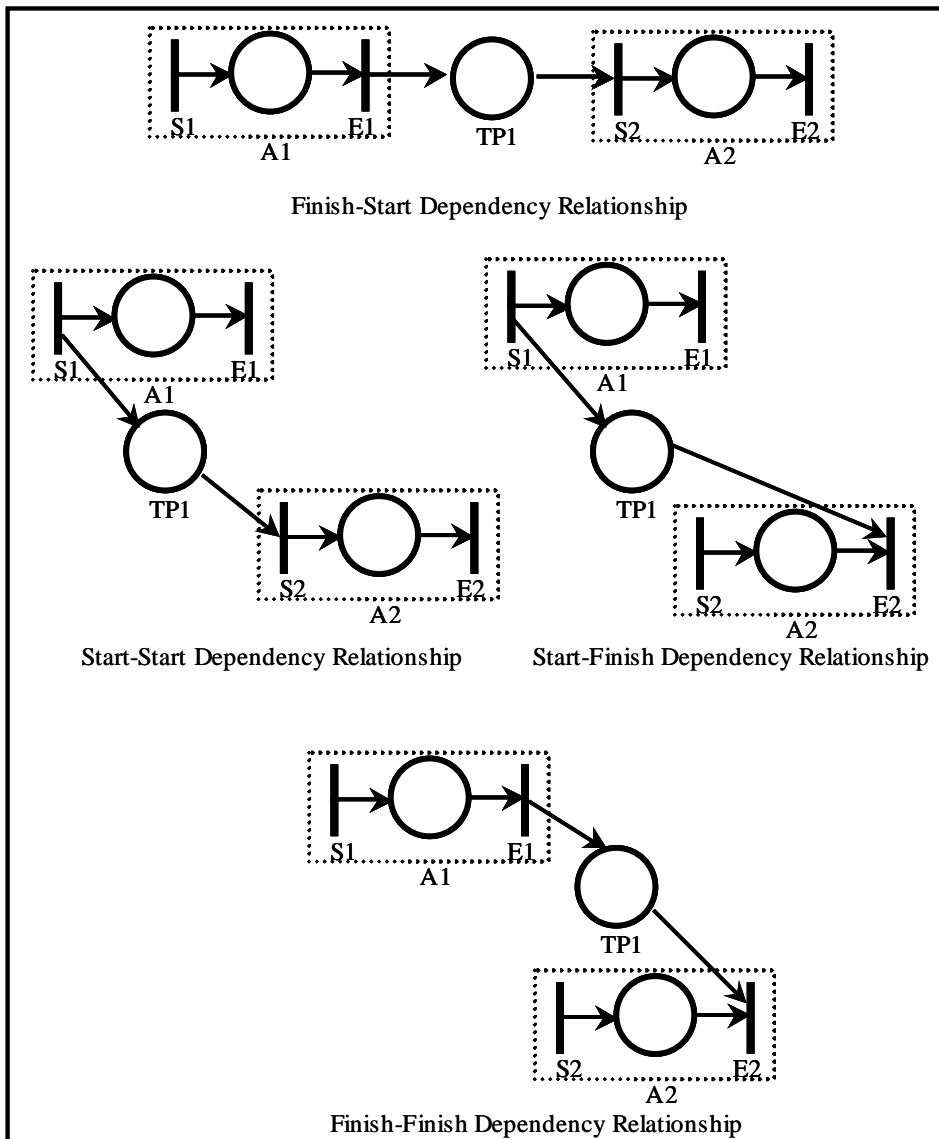


Fig. 4. Four types of activity dependency relationships.

CPM-D facilitates concurrent modeling of resources, tasks and products of a software development project. In the beginning, CPM-D can be used to simulate the project and estimate the expected quality and time span of the project. As the project proceeds, the actual quality of finished products and duration of finished activities and changes to resource tokens due to resource allocation across projects need to be updated and reflected. CPM-D can then be simulated again for the new scenario to dynamically provide responsive feedback in support of concurrent project management.

3.3 An Illustrative Example

In this section, a simple example is explained in detailed to illustrate the application of the proposed CPM-M. Based on the concept of CISF, the project manager analyzes all the entities discovered early in the project planning stage. This project is assumed to have four activities: design (A1), interface coding (A2), system coding (A3) and system integration (A4). Activities A2 and A3 can be performed concurrently. There are five kinds of products which are system analysis document (PP1), system design document (PP2), interface code (PP3), system code (PP4) and integrated code (PP5). Three kinds of human resources are required by activities in the activity pool and include designer (RP1), coding engineer (RP2), and integration engineer (RP3). Assume that each person belongs to only one type of resource. All the activities are connected with finish-start relationships among TP1, TP2, TP3, TP4, and TP5.

As for the resources, assume that a designer, three coding engineers and a system integration engineer are available. Hence this project can be statically modeled as shown in Fig. 5. Tokens C1 and P1 are obtained from the analysis stage. The details of all components are listed in Tables 2-6.

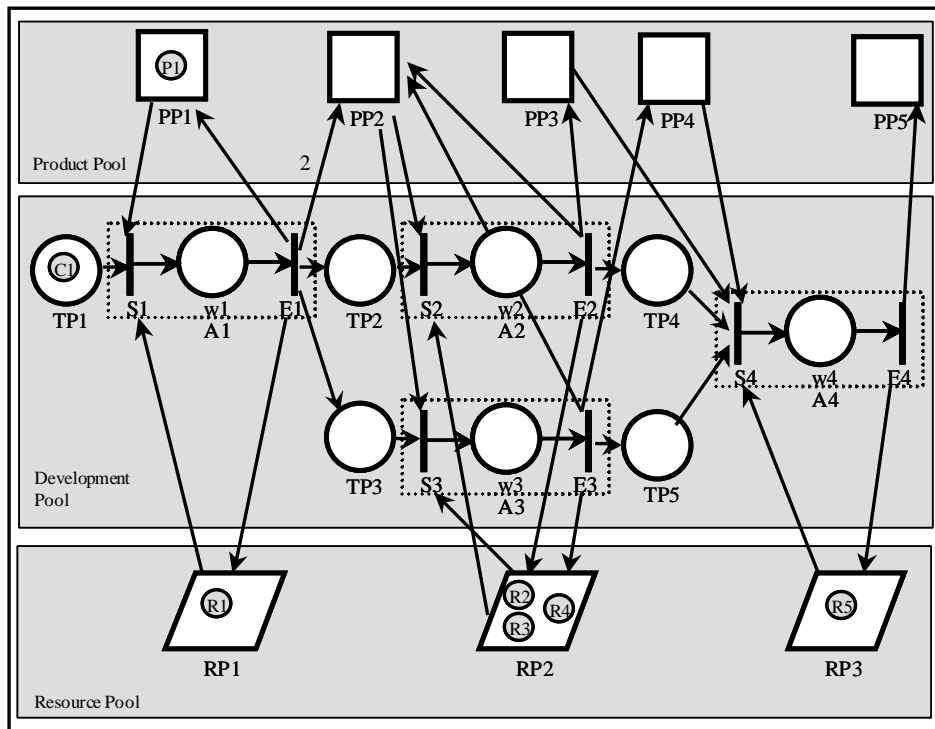


Fig. 5. Modeling of the illustrative example with CPM-D.

Table 2. Product places generated in the illustrative example.

ID	Type(Reference/ Semi-product)	Specification	Allocation strategy	Priority of Output Arc
PP1	R	system analysis document	Most fit	{{(PP1-S1)}}
PP2	R	system design document	Most fit	{{(PP2-S2), (PP2-S3)}}
PP3	S	interface code	Most fit	{{(PP3-S4)}}
PP4	S	system code	Most fit	{{(PP4-S4)}}
PP5	S	integrated code		

Table 3. Resource places generated in the illustrative example.

ID	Type(Consumable/ Reusable)	Specification	Allocation Strategy	Priority of Output Arc
RP1	R	designer	Most fit	{{(RP1-S1)}}
RP2	R	coding engineer	Most fit	{{(RP2-S2), (RP2-S3)}}
RP3	R	integration engineer	Most fit	{{(RP3-S4)}}

Table 4. Activity-start transitions defined in the illustrative example.

ID	Requirement{place ID, count, constraint} constraint = lower Product Quality/ lower Resource Capability/ upper Task complexity level	Output {place ID, count}	Determine generation times of output token	Determine activity duration to activity token	Determine task complexity level of transition token
S1	{{(TP1, 1, A), (PP1, 1, B), (RP1, 1, B)}}	{{(W1, 1)}}	Maximum of all the input tokens' generation times.	$D = F(L_r, L_t, L_p)$	B
S2	{{(TP2, 1, A), (PP2, 1, B), (RP2, 1, B)}}	{{(W2, 1)}}	Maximum of all the input tokens' generation times.	$D = F(L_r, L_t, L_p)$	B
S3	{{(TP3, 1, A), (PP2, 1, B), (RP2, 1, B)}}	{{(W3, 1)}}	Maximum of all the input tokens' generation times.	$D = F(L_r, L_t, L_p)$	B
S4	{{(PP3, 1, B), (PP4, 1, B), (TP4, 1, A), (TP5, 1, A), (RP3, 1, B)}}	{{(W4, 1)}}	Maximum of all the input tokens' generation times.	$D = F(L_r, L_t, L_p)$	B

In this example, we assume that the human resource capability, product quality and task complexity all have three discrete levels A, B, and C. Further, we partition the task history into twenty-seven sample spaces with different combinations of resource capability, product quality, and task complexity, i.e., (A, A, A), (AA, B), (A, A, C), ..., (C, C, C).

Table 5. Activity-end transitions defined in the illustrative example.

ID	Requirement	Output{ place ID, count }	Determine generation times of output token	Determine product quality of output product token	Determine task complexity level of transition token
E1	{(W1, 1)}	{(PP1, 1), (RP1, 1), (PP2, 2), (TP2, 1), (TP3, 1)}	Max{(activity duration of activity token from working place to the end transition + generation time of activity token from working place to the end transition), (maximum of all the other generation times of tokens to the end transition)}	$Q = F(L_r, L_t, L_p)$	{(E1-TP2, A), (E1-TP3, B)}
E2	{(W2, 1)}	{(PP2, 1), (PP3, 1), (TP4, 1), (RP2, 1)}	Max{(activity duration of activity token from working place to the end transition + generation time of activity token from working place to the end transition), (maximum of all the other generation times of tokens to the end transition)}	$Q = F(L_r, L_t, L_p)$	(E2-TP4, B)
E3	{(W3, 1)}	{(PP2, 1), (PP4, 1), (TP5, 1), (RP2, 1)}	Max{(activity duration of activity token from working place to the end transition + generation time of activity token from working place to the end transition), (maximum of all the other generation times of tokens to the end transition)}	$Q = F(L_r, L_t, L_p)$	(E3-TP5, B)
E4	{(W4, 1)}	{(PP5, 1), (RP3, 1)}	Max{(activity duration of activity token from working place to the end transition + generation time of activity token from working place to the end transition), (maximum of all the other generation times of tokens to the end transition)}	$Q = F(L_r, L_t, L_p)$	

Table 6. Initial attribute values of tokens.

ID	Type (reference/semi-product/ (consumable/reusable)	Quality level/Capability level/ Task complexity level	Time of generation
P1	reference	B	0
C1		A	0
R1	reusable	A	0
R2	reusable	A	0
R3	reusable	B	0
R4	reusable	B	0
R5	reusable	B	0

Table 7. The expected values of activity duration and product quality.

A	$F(L_r, L_s, L_p)$	E_d	E_q	A	$F(L_r, L_s, L_p)$	E_d	E_q	A	$F(L_r, L_s, L_p)$	E_d	E_q	A	$F(L_r, L_s, L_p)$	E_d	E_q
A1	F(A, A, A)	10	A	A2	F(A, A, A)	6	A	A3	F(A, A, A)	5	A	A4	F(A, A, A)	8	A
A1	F(A, A, B)	11	B	A2	F(A, A, B)	5	B	A3	F(A, A, B)	6	B	A4	F(A, A, B)	9	B
A1	F(A, A, C)	12	C	A2	F(A, A, C)	6	C	A3	F(A, A, C)	7	C	A4	F(A, A, C)	10	C
A1	F(A, B, A)	9	A	A2	F(A, B, A)	5	A	A3	F(A, B, A)	4	A	A4	F(A, B, A)	5	A
A1	F(A, B, B)	10	B	A2	F(A, B, B)	6	B	A3	F(A, B, B)	5	B	A4	F(A, B, B)	6	B
A1	F(A, B, C)	11	C	A2	F(A, B, C)	7	C	A3	F(A, B, C)	6	C	A4	F(A, B, C)	7	C
A1	F(A, C, A)	7	A	A2	F(A, C, A)	4	A	A3	F(A, C, A)	3	A	A4	F(A, C, A)	2	A
A1	F(A, C, B)	8	B	A2	F(A, C, B)	5	B	A3	F(A, C, B)	4	B	A4	F(A, C, B)	3	B
A1	F(A, C, C)	9	C	A2	F(A, C, C)	6	C	A3	F(A, C, C)	5	C	A4	F(A, C, C)	4	C
A1	F(B, A, A)	12	B	A2	F(B, A, A)	8	B	A3	F(B, A, A)	7	B	A4	F(B, A, A)	10	B
A1	F(B, A, B)	13	B	A2	F(B, A, B)	9	B	A3	F(B, A, B)	8	B	A4	F(B, A, B)	11	B
A1	F(B, A, C)	14	C	A2	F(B, A, C)	10	C	A3	F(B, A, C)	9	C	A4	F(B, A, C)	12	C
A1	F(B, B, A)	11	B	A2	F(B, B, A)	7	B	A3	F(B, B, A)	6	B	A4	F(B, B, A)	7	B
A1	F(B, B, B)	12	B	A2	F(B, B, B)	8	B	A3	F(B, B, B)	7	B	A4	F(B, B, B)	8	B
A1	F(B, B, C)	13	C	A2	F(B, B, C)	9	C	A3	F(B, B, C)	8	C	A4	F(B, B, C)	9	C
A1	F(B, C, A)	9	B	A2	F(B, C, A)	6	B	A3	F(B, C, A)	5	B	A4	F(B, C, A)	4	B
A1	F(B, C, B)	10	B	A2	F(B, C, B)	7	B	A3	F(B, C, B)	6	B	A4	F(B, C, B)	5	B
A1	F(B, C, C)	11	C	A2	F(B, C, C)	8	C	A3	F(B, C, C)	7	C	A4	F(B, C, C)	6	C
A1	F(C, A, A)	14	C	A2	F(C, A, A)	10	C	A3	F(C, A, A)	9	C	A4	F(C, A, A)	12	C
A1	F(C, A, B)	15	C	A2	F(C, A, B)	11	C	A3	F(C, A, B)	10	C	A4	F(C, A, B)	13	C
A1	F(C, A, C)	16	C	A2	F(C, A, C)	12	C	A3	F(C, A, C)	11	C	A4	F(C, A, C)	14	C
A1	F(C, B, A)	13	C	A2	F(C, B, A)	9	C	A3	F(C, B, A)	8	C	A4	F(C, B, A)	9	C
A1	F(C, B, B)	14	C	A2	F(C, B, B)	10	C	A3	F(C, B, B)	9	C	A4	F(C, B, B)	10	C
A1	F(C, B, C)	15	C	A2	F(C, B, C)	11	C	A3	F(C, B, C)	10	C	A4	F(C, B, C)	11	C
A1	F(C, C, A)	11	B	A2	F(C, C, A)	8	B	A3	F(C, C, A)	7	B	A4	F(C, C, A)	6	B
A1	F(C, C, B)	12	B	A2	F(C, C, B)	9	B	A3	F(C, C, B)	8	B	A4	F(C, C, B)	7	B
A1	F(C, C, C)	13	C	A2	F(C, C, C)	10	C	A3	F(C, C, C)	9	C	A4	F(C, C, C)	8	C

A: Activity; E_d : Expected Value of Activity Duration; E_q : Expected Value of Product Quality

As in the Three-Estimate Approach of PERT [5], we assume the probability distributions of each activity duration and output quality are approximated by beta distributions to estimate the expected value of quality and duration in different sample spaces (Table 7).

Based on the above data, the manager can use the proposed CPM-M to estimate the predicted project finish time and product quality early in project planning stage. The key attributes of tokens that change over the simulation are described in Table 8. According to the transitions enabled, Table 8 shows how the transition tokens, resource tokens, product tokens and activity tokens changed in each place. The token specifications include (transition token id, time of generation, task complexity), (resource token id, time of generation, capability level), (product token id, time of generation, quality level), and (activity token id, time of generation, activity duration).

Table 8. Transitions enabled at specific times T0 ~ T7.

		TP1	TP2	TP3	TP4	TP5	W`1	W`2	W`3	W`4
T0	ini	(C1, 0, A)								
T1	S1						(A1, 0, 11)			
T2	E1		(C2, 11, B)	(C3, 11, B)						
T3	S2			(C3, 11, B)				(A2, 11, 8)		
	S3							(A2, 11, 8)	(A3, 11, 7)	
T4	E2				(C4, 19, B)				(A3, 11, 7)	
T5	E3				(C4, 19, B)	(C5, 18, B)				
T6	S4									(A4, 19, 8)
T7	E4									

		PP1	PP2	PP3	PP4	PP5	RP1	RP2	RP3
T0	ini	(P1, 0, B)					(R1, 0, A)	(R2, 0, A), (R3, 0, B), (R4, 0, B)	(R5, 0, B)
T1	S1							(R2, 0, A), (R3, 0, B), (R2, 0, B)	(R5, 0, B)
T2	E1	(P1, 11, B)	{(P2, 11, B), (P3, 11, B)}				(R1, 11, A)	(R2, 0, A), (R3, 0, B) (R2, 0, B)	(R5, 0, B)
T3	S2	(P1, 11, B)	(P3, 11, B)				(R1, 11, A)	(R2, 0, A), (R4, 0, B)	(R5, 0, B)
	S3	(P1, 11, B)					(R1, 11, A)		(R5, 0, B)
T4	E2	(P1, 11, B)	(P2, 19, B)	(P4, 19, B)			(R1, 11, A)	(R2, 0, A), (R3, 19, B)	(R5, 0, B)
T5	E3	(P1, 11, B)	{(P2, 19, B), (P3, 18, B)}	(P4, 19, B)	(P5, 18, B)		(R1, 11, A)	(R2, 0, A), (R3, 19, B), (R4, 18, B)	(R5, 0, B)
T6	S4	(P1, 11, B)	{(P2, 19, B), (P3, 18, B)}				(R1, 11, A)	(R2, 0, B), (R3, 19, B), (R4, 18, B)	
T7	E4	(P1, 11, B)	{(P2, 19, B), (P3, 18, B)}			(P6, 27, B)	(R1, 11, A)	(R2, 0, B), (R3, 19, B), (R4, 18, B)	(R5, 27, B)

All transitions are enabled at eight different time points from T0 to T7 (see Table 9). Initial time is given by the manager and other are determined according to input tokens and other requirements. The product quality levels that are determined by product and resource tokens, are presented in Table 10.

Table 9. Times determined over simulation.

		Determined by input tokens	Time
T0	ini		0
T1	S1	{{(C1, 0, A), (P1, 0, B), (R1, 0, A)}}	0
T2	E1	{{(A1, 0, 11)}}	11
T3	S2	{{(C2, 11, B), (P2, 11, B), (R3, 0, B)}}	11
	S3	{{(C3, 11, B), (P3, 11, B), (R4, 0, B)}}	11
T4	E2	{{(A2, 11, 8)}}	19
T5	E3	{{(A3, 11, 7)}}	18
T6	S4	{{(C4, 19, B), (C5, 18, B), (P4, 19, B), (P5, 18, B), (R5, 0, B)}}	19
T7	E4	{{(A4, 19, 8)}}	27

Table 10. Output product quality levels determined.

		Determined by Product tokens and Resource tokens	Quality
T2	E1	{{(P1, 0, B), (R1, 0, A)}}	B
T4	E2	{{(P2, 11, B), (R3, 0, B)}}	B
T5	E3	{{(P3, 11, B), (R4, 0, B)}}	B
T7	E4	{{(P4, 19, B), (P5, 18, B), (R5, 0, B)}}	B

From the complete simulation process, the proposed method can help the manager plan the reasonable duration of each activity and estimate levels the product quality of all products (see Table 10). Especially when many parallel activities are performed with limited available resources and products, it is critical for the manager to assign resources and products to activities according to appropriate allocation strategy. The project planning can hence achieve better performance in terms of project time span and quality from CE perspective.

Here we describe how the product or resource allocation strategy will affect the results of a project. In this example, we use a “most fit” strategy that allocates the satisfied resource or product closest to the constraint. If we change the strategy on “RP2” to “best fit” strategy, which allocates the best level of product or resource available, the duration of “best fit” strategy will be 1 unit longer/shorter than “most fit” strategy while the project quality remains the same. In other words, the proposed approach can support the manager in analyzing the selection of strategy early in planning stage. It must be emphasized that, according to the relation of cost, performance and duration of the project, the strategy that is best can be determined by the manager based on the objectives of the pro-

ject management. For instance, if the objective of the project management is to maximize customer satisfaction, one can adopt the “best fit” strategy. If the objective of the project management is to minimize the cost, one may consider using the “most fit” strategy. More strategies to allocate the resources and products can be defined according to the requirements and goals of the project management. In addition, the CPM-D result varies depending on the decisions made, such as the human resource allocation which may lead to poor product quality and hence additional rework. In this illustrative example, CPM-M has been used to plan and simulate a project and help achieve better performance. To better demonstrate the applicability and scalability of CPM-M, a case study was conducted as discussed in section 4.

4. CASE STUDY

The proposed CPM-M was applied to and simulated a real case, which was a information system development project of a web-based campaign system for the marketing division of a bank. The development teams of various specialties have been trained to develop systems based on *web-based system development templates*, which include system design, coding, testing, etc.

4.1 Development of a Web-Based Campaign System

Before applying CPM-M, the development process of the campaign system was carefully investigated. In the bank, many teams work on multiple projects concurrently, and therefore effective scheduling is critical because resource conflict occurs frequently during the development process. In addition, the bank has severe quality/reliability requirements for the systems, due to the heavy loading, 24-hour service, and high standard on data security.

The major process of developing the web-based campaign system is described as follows. After the project manager is assigned a project, the system requirements and scope are soon defined. System and architecture analyses then proceed concurrently. Once the analyses are completed, user interface layer, business layer, and system layer design, coding and testing will start. In the meantime, the hardware purchase process is initiated and a testing case design is carried out. The results of the testing case design will be referenced by the design, coding and testing layers. The complete user interface designs will then be integrated and validated before the editing of system training document can start. The design, coding and testing layers, along with the purchased hardware are then integrated. The integrated system thereafter has to pass through system level testing and debugging before going into production. The experiences and feedback from the pilot production runs of the system will provide guidelines for successive system validation and modification. Soon after the users are trained, the project is closed.

4.2 CISF

The first component of CPM-M is the conceptual CISF. With such an information sharing platform, all related data about the project can be shared with the development

teams throughout the project development life cycle. The critical data are categorized into three information pools, activity, resource and product pools:

1. Activity pool: Based on the systems requirements and the results of system and architecture analyses, processes were decomposed into activities as shown in Table 11. The activity complexity can be defined as a function of task template. In the case study, for the task template of “function,” the possible domain values are {A, B, C}, where A indicates that the customized function has the highest complexity level, and B and C indicate mixed and standard functions, respectively. For the task template “topic,” the complexity domain contains values A and B, which refer to OLAP and OLTP operations on the data, respectively. For the other task templates, Table 11 gives the complexity levels. The duration of an activity has been defined as $D = F(L_r, L_t, L_p)$ in section 3.2. For instance, an empirical testing and debugging activity duration may be estimated as $TDD = \text{number of functions} * \text{Average time per function} * k_{td}$, where k_{td} is a test and debugging scaling constant. Similarly, an empirical system integration activity duration may be estimated as $SID = \text{number of functions} * \text{average time per function} * k_{si}$, where k_{si} is a system integration scaling constant; and an empirical training activity duration may be estimated as $TD = \text{number of use cases} * \text{average time per use case} * k_t$, where k_t is a training scaling constant.

Table 11. Activities generated in the case study.

No	Activity Name	Average Complexity	Complexity Domain	Task Template
A1	Requirements and scope	A	(A)	Requirement
A2	System Analysis	A	(A)	Analysis
A3	System Architecture Analysis	A	(A)	Architecture
A4	User Interface Layer Design, Coding & Testing	B	(A, B, C)	Function
A5	Business Layer Design, Coding & Testing	A	(A, B)	Topic
A6	System Layer Design, Coding & Testing	B	(A, B, C)	Function
A7	Hardware Purchasing	A	(A)	Order
A8	Testing Case Design	A	(A)	Use Case
A9	User Interface Integration & Validation	A	(A)	Use Case
A10	System Integration	A	(A)	Function
A11	Editing System Training Document	A	(A)	Use Case
A12	Hardware Planning and Setup	B	(A, B, C)	Hardware
A13	System Testing & Debugging	A	(A)	Use Case
A14	System Going Production	A	(A)	Implement
A15	System Validation & Modification	A	(A)	Use Case
A16	System User Training	A	(A)	Use Case
A17	Project Close	A	(A)	Project

2. Resource pool: Given the human resources of various specialties assigned to the project, seven resource places were generated for the case study as shown in Table 12.

Table 12. Resource places generated in the case study.

ID	Type(Consumable/Reusable)	Specification	Allocation Strategy	Priority of Output Arc
RP1	R	Project Manager	Most Fit	{(R1-S1), (R1-S17)}
RP2	R	System Analyst	Most Fit	{(R2-S1), (R2-S2), (R2-S8), (R2-S13), (R2-S15), (R2-S17)}
RP3	R	Architect	Most Fit	{(R3-S1), (R3-S3), (R3-S7), (R3-S12), (R3-S10), (R3-S15), (R3-S17)}
RP4	R	Facilitate Management Engineer	Most Fit	{(R4-S12), (R4-S10)}
RP5	R	User Interface Layer Designer	Most Fit	{(R4, S4), (R4-S9), (R4-S11), (R4-A13), (R4-A16)}
RP6	R	Business Layer Designer	Most Fit	{(R5-S5), (R5-S9), (R5-S11), (R5-A13), (R5-A15)}
RP7	R	System Layer Designer	Most Fit	{(R6-S4), (R6-S9), (R6-S11), (R6-A13), (R5-A15)}

3. Product pool: Based on the system requirements and generated activities, 22 product places were generated in the case study as shown in Table 13.

With all three pools of information identified and shared via CISF across different project activities, concurrent project development can be promoted.

4.3 Constructing CPM-D

After identifying all entities of the activity, resource and product pools, we started to construct the CPM-D for the web-based campaign system. Since there were 17 activities designated, the corresponding activity-start and activity-end transitions were defined and given in Tables 14 and 15, respectively. Table 16 lists tokens assigned in Product, Resource and Transition places in the CPM-D when Activity 3 is completed. The reason why the initial state of CPM-D is defined after Activity 3 is completed is because we cannot have overall and complete understanding of all the components without the completion of Activities 1 to 3. The constructed CPM-D for the case study is shown in Fig. 6, where relations between activity pool and other pools are omitted.

Table 13. Product places generated in the case study.

ID	Type(Reference/ Semi-product)	Specification	Allocation Strategy	Priority of Output Arc
PP1	R	Requirement Specification	Most Fit	{{(P1-S2), (P1-S3)}}
PP2	R	System Analysis Specification	Most Fit	{{(P2-S3), (P2, S4), (P2, S5), (P2-S6), ..., (P2-S17)}}
PP3	R	System Architecture Specification	Most Fit	{{(P3, S4), (P3-S5), (P3-S6), ..., (P3-S17)}}
PP4	R	System Interface Layer Design & Testing Specification	Most Fit	{{(P4-S9), (P4-S10)}}
PP5	R	Business Layer Design & Testing Specification	Most Fit	{{(P5-S10)}}
PP6	R	System Layer Design & Testing Specification	Most Fit	{{(P6-S10)}}
PP7	R	Interface Layer Unit Integration and Validation Report	Most Fit	
PP8	R	Hardware Planning and Setup Report	Most Fit	{{(P8-S10)}}
PP9	R	System Integration Report	Most Fit	
PP10	R	System Testing Case Specification	Most Fit	{{(P10-S13)}}
PP11	R	System Testing & Modify Report	Most Fit	
PP12	R	System Install Report	Most Fit	
PP13	R	User Validation and System Modify Report	Most Fit	
PP14	R	System Training Document	Most Fit	{{(P14-S16)}}
PP15	R	Project Close Report	Most Fit	
PP16	S	User Interface Layer Unit Code	Most Fit	{{(P16-S9)}}
PP17	S	Integrated & Validated Interface Code	Most Fit	{{(P17-S10)}}
PP18	S	Business Layer Unit Code	Most Fit	{{(P18-S10)}}
PP19	S	System Layer Unit Code	Most Fit	{{(P19-S10)}}
PP20	S	Integrated System Program	Most Fit	{{(P20-S13)}}
PP21	S	Tested System Program	Most Fit	{{(P21-S14)}}
PP22	S	Validated System Program	Most Fit	

Table 14. Activity-start transitions defined in the case study.

ID	Requirement {place ID, count, constraint}	Output {place ID, count }	Determine generation times of output token	Determine activity duration to activity token	Determine task complexity level of transition token
S1	{(TP0,1,A),(RP1,1,A),(RP2,1,A),(RP3,1,A)}	{(W1,1)}	Maximum of all the input tokens' generation times.	20(hours)	A
S2	{(TP1,1,A),(RP2,1,A),(PP1,1,C)}	{(W2,1)}	Maximum of all the input tokens' generation times.	25(hours)	A
S3	{(TP2,1,A),(TP3,1,A),(RP3,1,A),(PP1,1,C),(PP2,1,C)}	{(W3,1)}	Maximum of all the input tokens' generation times.	30(hours)	A
S4	{(TP4,1,A),(RP5,1,C),(PP2,1,C),(PP3,1,C)}	{(W4,1)}	Maximum of all the input tokens' generation times.	$D=F(L_{\tau}, L_{\tau}, L_{\tau})$	Max(input transition token complexity)
S5	{(TP5,1,A),(RP6,1,B),(PP2,1,C),(PP3,1,C)}	{(W5,1)}	Maximum of all the input tokens' generation times.	$D=F(L_{\tau}, L_{\tau}, L_{\tau})$	Max(input transition token complexity)
S6	{(TP6,1,A),(RP7,1,C),(PP2,1,C),(PP3,1,C)}	{(W6,1)}	Maximum of all the input tokens' generation times.	$D=F(L_{\tau}, L_{\tau}, L_{\tau})$	Max(input transition token complexity)
S7	{(TP7,1,A),(RP3,1,B),(PP3,1,C)}	{(W7,1)}	Maximum of all the input tokens' generation times.	5(hours)	A
S8	{(TP8,1,A),(RP2,1,B),(PP2,1,C),(PP3,1,C)}	{(W8,1)}	Maximum of all the input tokens' generation times.	6(hours)	A
S9	{(TP9,Number of Function,A),(PP2,1,C),(PP3,1,C),(RP5,1,C),(PP6,1,C),(PP16,Number of Function)}	{(W9,1)}	Maximum of all the input tokens' generation times.	$D=F(L_{\tau}, L_{\tau}, L_{\tau})$	B
S10	{(TP10,1,A),(TP11,1,A),(TP12,1,A),(TP15,1,A),(RP3,1,B),(RP4,1,C),(PP2,1,C),(PP3,2,C),(PP17,1,C),(PP18,1,C),(PP19,1,C)}	{(W10,1)}	Maximum of all the input tokens' generation times.	3(hours)	A
S11	{(TP13,1,A),(RP5,1,C),(PP2,1,C),(PP3,1,C),(PP4,1,C)}	{(W11,1)}	Maximum of all the input tokens' generation times.	3(hours)	A
S12	{(TP14,1,A),(RP3,1,C),(RP4,1,C),(PP3,1,B)}	{(W12,1)}	Maximum of all the input tokens' generation times.	$D=F(L_{\tau}, L_{\tau}, L_{\tau})$	Max(input transition token complexity)
S13	{(TP16,1,A),(TP17,1,A),(RP2,1,B),(RP3,1,B),(RP5,1,C),(RP6,1,B),(RP7,1,C),(PP2,1,C),(PP3,1,C),(PP10,1,C),(PP20,1,C)}	{(W13,1)}	Maximum of all the input tokens' generation times.	$D=F(L_{\tau}, L_{\tau}, L_{\tau})$	A
S14	{(TP18,1,A),(RP7,1,C),(PP3,1,B),(PP21,1,C)}	{(W14,1)}	Maximum of all the input tokens' generation times.	$D=F(L_{\tau}, L_{\tau}, L_{\tau})$	A
S15	{(TP19,1,A),(RP2,1,B),(RP3,1,B),(PP21,1,C)}	{(W15,1)}	Maximum of all the input tokens' generation times.	$D=F(L_{\tau}, L_{\tau}, L_{\tau})$	A
S16	{(TP20,1,A),(TP21,1,A),(RP5,1,A),(PP14,1,C)}	{(W16,1)}	Maximum of all the input tokens' generation times.	$D=F(L_{\tau}, L_{\tau}, L_{\tau})$	A
S17	{(TP22,1,A),(RP1,1,A),(RP2,1,A),(PP1,1,C),(PP2,1,C),(PP3,1,C)}	{(W17,1)}	Maximum of all the input tokens' generation times.	3(hours)	A

Table 15. Activity-end transitions defined in the case study.

ID	Requirement	Output	Determine generation times of output token	Determine product quality of output product token	Determine task complexity level of transition token
E1	{{(W1,1)}	{{(PP1,1),(RP1,1),(RP2,1),(RP3,1),(TP1,1),(TP2,1)}	Max{ (activity duration of activity token from working place into the end transition + generation time of activity token from working place into the end transition), (maximum of all the other generation times of tokens into the end transition)}	A	{{(E1-TP2,A)}
E2	{{(W2,1)}	{{(PP2,1),(PP1,1),(RP2,1),(TP3,1),(TP8,1)}	formula is the same as E1	A	{{(E2-TP,B)}
E3	{{(W3,1),(TP3,1),(PP2,1)}	{{(PP3,1),(PP1,1),(PP2,1),(RP3,1),(TP4,number of business function),(TP5,number of topic),(TP6,number of system function)}	formula is the same as E1	$Q=F(L_r, L_t, L_p)$	{{(E3-TP4, B),(E3-TP5,A),(E3-TP6,B)}
E4	{{(W4,1)}	{{(TP9,1),(PP4,1),(PP16,1),(PP2,1),(PP3,1),(RP5,1)}	formula is the same as E1	$Q=F(L_r, L_t, L_p)$	{{(E4-TP9,A)}
E5	{{(W5,1)}	{{(TP10,1),(PP5,1),(PP18,1),(RP6,1)}	formula is the same as E1	$Q=F(L_r, L_t, L_p)$	{{(E5-TP10,A)}
E6	{{(W6,1)}	{{(TP11,1),(PP2,1),(PP3,1),(PP6,1),(PP19,1),(RP7,1)}	formula is the same as E1	$Q=F(L_r, L_t, L_p)$	{{(E6-TP11,A)}
E7	{{(W7,1)}	{{(PP3,1),(RP3,1),(TP14,Number of Hardware)}	formula is the same as E1	$Q=F(L_r, L_t, L_p)$	{{(E7-TP14,B)}
E8	{{(W8,1)}	{{(PP2,1),(PP3,1),(RP2,1),(TP14,Number of UseCase)}	formula is the same as E1	$Q=F(L_r, L_t, L_p)$	{{(E8-TP17,A)}
E9	{{(W9,1)}	{{(RP5,1),(PP2,1),(PP3,1),(PP7,1),(PP17,1),(TP12,1),(TP13,1)}	formula is the same as E1	$Q=F(L_r, L_t, L_p)$	{{(E9-TP12,A),(E9-TP13,A)}
E10	{{(W10,1)}	{{(TP16,1),(RP2,1),(RP3,1),(RP4,1),(PP2,1),(PP3,1),(PP20,1)}	formula is the same as E1	$Q=F(L_r, L_t, L_p)$	{{(E10-TP16,A)}
E11	{{(W11,1)}	{{(RP5,1),(PP2,1),(PP3,1),(PP4,1),(PP14,1),(TP20,1)}	formula is the same as E1	$Q=F(L_r, L_t, L_p)$	{{(E11-TP20,A)}
E12	{{(W12,1)}	{{(TP15,1),(RP3,1),(RP4,1),(PP3,1,B),(RP8,1)}	formula is the same as E1	$Q=F(L_r, L_t, L_p)$	{{(E12-TP15,A)}
E13	{{(W13,1)}	{{(RP2,1),(RP3,1),(RP5,1),(RP6,1),(PP2,1),(PP3,1),(PP10,1),(PP21,1),(TP18,1)}	formula is the same as E1	$Q=F(L_r, L_t, L_p)$	{{(E13-TP18,A)}
E14	{{(W14,1)}	{{(TP19,1),(RP7,1),(PP3,1),(PP21,1),(PP12,1)}	formula is the same as E1	$Q=F(L_r, L_t, L_p)$	{{(E14-TP19,A)}
E15	{{(W15,1)}	{{(TP21,1),(RP2,1),(RP3,1),(PP22,1)}	formula is the same as E1	$Q=F(L_r, L_t, L_p)$	{{(E15-TP21,A)}
E16	{{(W16,1)}	{{(TP22,1),(RP5,1),(PP14,1)}	formula is the same as E1	$Q=F(L_r, L_t, L_p)$	{{(E16-TP22,A)}
E17	{{(W17,1)}	{{(RP2,1),(RP1,1),(RP2,1),(PP1,1),(PP2,1),(PP3,1)}	formula is the same as E1	$Q=F(L_r, L_t, L_p)$	

Table 16. Tokens assigned in product, resource and transition places when activity 3 is completed.

Place ID	Token ID	Type (reference/ semi-product)/ (consumable/reusable)	Quality level/ Capability level/ Task complexity level	Time of generation	Number of Copies of Reference Product
PP1	P1	reference	A	50	17
PP2	P2	reference	A	50	17
PP3	P3	reference	A	50	17
TP4	C4		B	50	
TP4	C5		B	50	
TP4	C6		B	50	
TP4	C7		A	50	
TP4	C8		B	50	
TP4	C9		C	50	
TP4	C10		C	50	
TP4	C11		C	50	
TP4	C12		B	50	
TP4	C13		B	50	
TP4	C14		B	50	
TP4	C15		B	50	
TP4	C16		B	50	
TP4	C17		B	50	
TP4	C18		B	50	
TP4	C19		B	50	
TP4	C20		B	50	
TP4	C21		B	50	
TP4	C22		B	50	
TP4	C23		B	50	
TP4	C24		B	50	
TP5	C25		A	50	
TP5	C26		B	50	
TP5	C27		A	50	
TP5	C28		A	50	
TP5	C29		A	50	
TP6	C30		B	50	
TP6	C31		B	50	
TP6	C32		B	50	
TP7	C33		A	50	
TP8	C34		A	50	
RP1	R1	reusable	A	50	
RP2	R2	reusable	A	50	
RP2	R3	reusable	B	50	
RP3	R4	reusable	A	50	
RP3	R5	reusable	B	50	
RP5	R6	reusable	A	50	
RP5	R7	reusable	C	50	
RP6	R8	reusable	A	50	
RP6	R9	reusable	A	50	
RP6	R10	reusable	B	50	
RP7	R11	reusable	A	50	
RP7	R12	reusable	C	50	

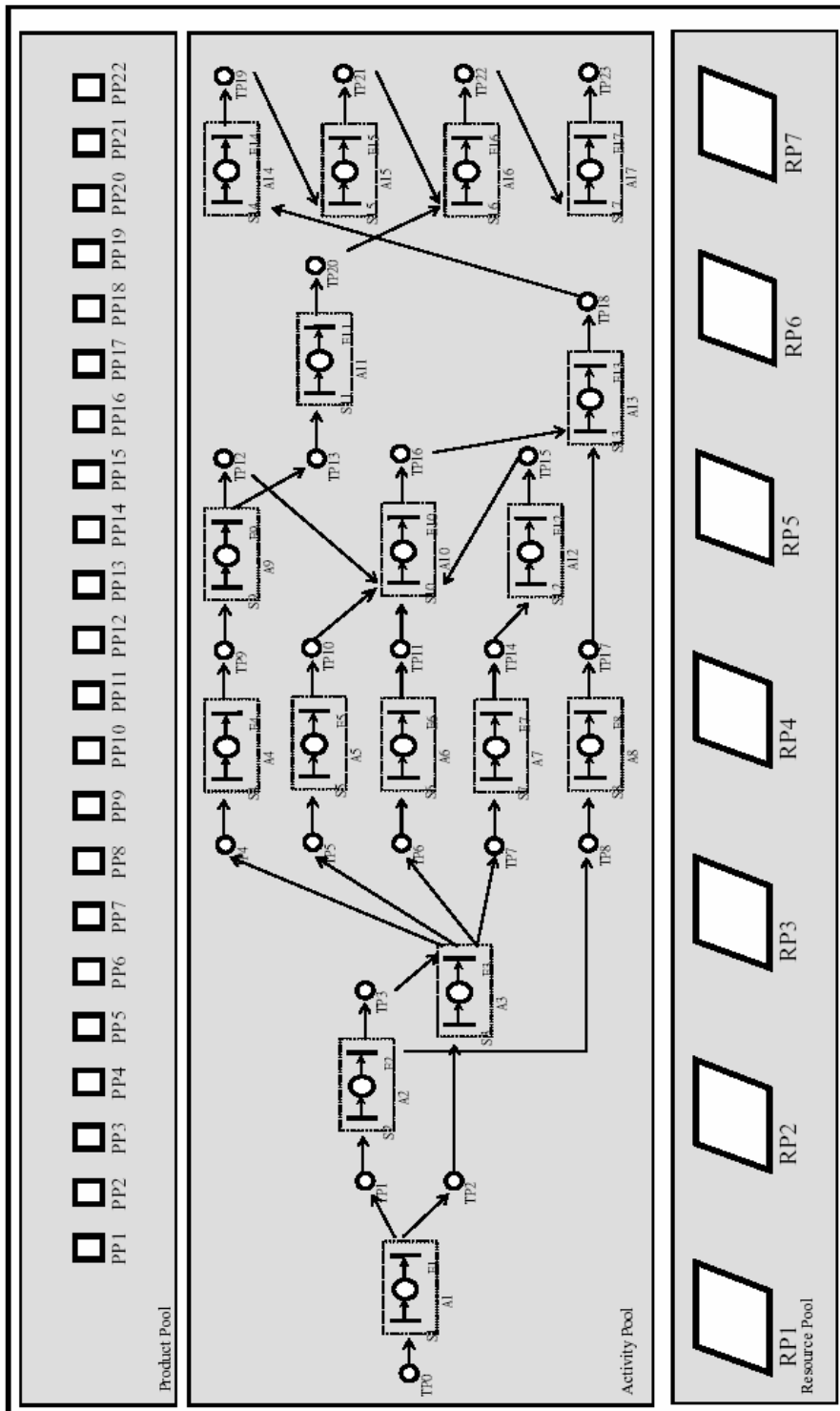


Fig. 6. Modeling of the case study with CPM-D with relations between activity pool and other pools omitted.

4.4 Case Findings

In this section two scenarios were simulated to demonstrate the capability of the proposed CPM-M. The project manager, based on his experience, estimated that the software project will be completed in approximately 54 days, of which 40 days are for analysis, design and coding, and 14 for testing, debugging, etc. The constructed CPM-D was simulated for the same human resource assignment as that by the project manager. The estimated time required to complete the project was about 70 days, which is quite longer than the project manager's professional judgment. After investigation, it was found that the difference is due to the assignment of a less experienced system designer. For more complicated OLAP data manipulation, if a less experienced less capable designer is assigned, the time required for the design process may be lengthened and the resultant product quality becomes unsatisfactory. Poor design quality, furthermore, will lead to delay in testing and modification activities. The above finding shows the potential advantage of the proposed CPM-M – considering all three pools of data, i.e., activity, resource and product data simultaneously, may result in better collaboration and more accurate project management.

Another circumstance was simulated when the project had been underway for 7 days, it was found that the less experienced designer has delayed his jobs and the product quality was unacceptable. The project manager decided to replace the designer with a more experienced designer. The CPM-D was rescheduled and the new estimated time to complete the project was 60 days, including the 7 days. It was then recommended to add another designer so that the design activities no longer become the bottleneck. Then the result of another round of CPM-D simulation turned out to be 52 days. Another problem occurred afterward, that the engineer taking charge of hardware planning and setup was temporarily unavailable because he was occupied with another ongoing project. After the constraint was considered in the CPM-D, the rescheduling result showed that the estimated time to complete the project became 58 days. The circumstance described here demonstrates CPM-M's potential capability to dynamically estimate the schedule and quality.

The case study also raises some of the interesting topics in software project management. One is the common resource conflict problem for multiple projects management, where we can construct an activity and a product pool for each project, and where multiple activity and product pools share the same and only resource pool. This way, resource allocation among multiple projects can be better depicted and simulated. Another issue is that intermittent resource availability may complicate the project scheduling and resource assignment problems. Without a concurrent project development and advanced scheduling algorithm, these issues are not trivial.

5. CONCLUSIONS

To realize concurrent engineering in software project management, we must have a suitable and practical method to support the project manager in cross-functional integration and decision making. In the cross-functional integration, which is based on the conceptual framework CISF, the CPM-D can support a manager to model a software project

and estimate the project schedule and product quality early in the project planning stage. The CPM-D can also be applied repeatedly at different times during the software development process so as to continuously monitor and provide responsive feedback for ensuring good project results.

There are three advantages of using the proposed CPM-M. Firstly, the CISF captures the three information pools in the software development process and CPM-D can simulate and visualize the project detail and is easy to apply. Secondly, CPM-M, by sharing information throughout the project development life cycle, breaks the cross-functional boundary and thus enhances concurrent cross-functional integration. Thirdly, the CPM-M empowers the project manager to use it for static planning of the project and dynamic estimation of duration, quality, project feasibility, etc.

In the future, more efficient resource and product allocation strategies or algorithms can be developed, and the CPM-M can also be computerized with a user friendly interface. Via network and database technology, product, activity, and resource information can be easily shared, described, and controlled by the personnel involved and the project manager. After being computerized as a project management tool, it not only can help to plan and estimate the project result early in the project planning stage but also can monitor the project progress and dynamically adjust the project with real feedback at any time. Finally, it will also be challenging to improve CPM-M by embedding a scheduling algorithm, such as critical chain, to facilitate concurrent scheduling and management of multiple parallel projects.

REFERENCES

1. M. Aoyama, "Management of distributed concurrent development for large-scale software systems," in *Proceedings of Asia Pacific Software Engineering Conference (APSEC '95)*, 1995, pp. 158-167.
2. A. Kumar and L. S. Ganesh, "Use of Petri nets for resource allocation in projects," *IEEE Transactions on Engineering Management*, Vol. 45, 1998, pp. 49-56.
3. C. Chang and M. Christensen, "A net practice for software project management," *IEEE Software*, Vol. 16, 1999, pp. 80-88.
4. C. W. Dawson, "Generalise activity on the node networks for managing uncertainty in project," *International Journal of Project Management*, Vol. 13, 1995, pp. 353-362.
5. E. A. Elsayed and T. O. Bouchen, *Analysis and Control of Production Systems*, Prentice-Hall, 1985.
6. J. E. Hallows, *Information Systems Project Management*, AMACOM, 1998.
7. F. S. Hillier and G. J. Lieberman, *Introduction to Operations Research*, McGraw-Hill, 1995.
8. H. Kerzner, *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*, ITP, 1995.
9. J. P. Lewis, *Mastering Project Management*, McGraw-Hill, 1998.
10. J. P. Lewis, *Project Planning Scheduling and Control*, McGraw-Hill, 1995.
11. L. C. Liu and E. Horwitz, "A formal model for software project management," *IEEE Transactions on Software Engineering*, Vol. 15, 1989, pp. 1280-1293.

12. S. S. Medhat and J. L. Rook, "Concurrent engineering-processes and techniques for the agile manufacturing enterprise," *The 5th International Conference on Factory 2000*, Vol. 2, 1997, pp. 9-14.
13. T. Murata, "Petri nets: properties, analysis and applications," in *Proceedings of the IEEE*, Vol. 77, 1989, pp. 541-580.
14. A. C. Nelson and K. Joshi, "Application of a matrix approach to estimate project skill requirements," *Information and Management*, Vol. 29, 1995, pp. 165-172.
15. A. Noore, "Electronic product design and manufacture in a concurrent engineering environment," *IEEE Transactions on Consumer Electronics*, Vol. 38, 1992, pp. 666-670.
16. P. O'Grady and J. S. Oh, "A review of approaches to design for assembly," *Concurrent Engineering*, Vol. 1, 1991, pp. 5-11.
17. F. Rafii and S. Perkins, "Internationalizing software with concurrent engineering," *IEEE Software*, Vol. 12, 1995, pp. 39-46.
18. J. Pennell and R. Winner, "Concurrent engineering practices and prospects," in *Proceedings of IEEE Global Telecommunications Conference and Exhibition (GLOBECOM '89)*, 1989, pp. 647-655.
19. A. Sawhney, "Petri net based simulation of construction schedules," in *Proceedings of Winter Simulation Conference*, 1997, pp. 1111-1118.
20. H. Steyn, "An investigation into the fundamentals of critical chain project scheduling," *International Journal of Project Management*, Vol. 19, 2001, pp. 363-369.
21. M. L. Swink, "A tutorial on implementing concurrent engineering in new product development programs," *Journal of Operation Management*, Vol. 16, 1998, pp. 103-116.
22. R. I. Winner, "The role of concurrent engineering in weapons systems acquisition," IDA Report R-338, Institute of Defense Analysis, 1988.
23. Z. Zhiwei and B. H. Ronald, "A simplified method of evaluating PERT/CPM network parameters" *IEEE Transactions on Engineering Management*, Vol. 41, 1994, pp. 426-430.



Jau-Ji Shen (沈肇基) received Ph.D. degree in Information Engineering and Computer Science from National Taiwan University at Taipei in 1988. From 1988 to 1994, he was the leader of the software group in Institute of Aeronautic, Chung-Sung Institute of Science and Technology. The following nine years after 1994, he was an associate professor of Information Management Department in the Chaoyang University of Technology at Taichung. He is currently an Associate Professor of Information Management Department in the National Formosa University at Yunlin. His current research areas focus on the data engineering, database techniques, and information security.



S. Wesley Changchien (張簡尚偉) is an Associate Professor with the Institute of Electronic Commerce at National Chung Hsing University, Taiwan, R.O.C. He received a B.S. degree in Mechanical Engineering (1989) and completed his M.S. (1993) and Ph.D. (1996) degrees in Industrial Engineering at State University of New York at Buffalo, U.S.A. His current research interests include electronic business management, internet/database marketing, knowledge management, data mining, and decision support systems.



Tao-Yuan Lin (林道元) is a system analyst of Information Technology Department at First Commercial Bank. He received the B.A. (1998) and M.B.A (2000) degrees in Information Management at Chaoyang University of Technology, Taiwan, R.O.C. His current research interests include business intelligence, customer relationship management, data warehouse, database marketing, and object-oriented technology.