Automatic Creation of a Crashing-Based Schedule Plan As Countermeasures against Process Delay

Daisuke Kinoshita, Rhito Yaegashi, Kazuhiro Uenosono,
Hiroaki Hashiura, Hiroki Uchikawa, and Seiichi Komiya

***Abstract*—** Development of large-scale software is usually conducted through a project to unite a work force. In addition, no matter what kind of life cycle model is adopted, a development plan is required for a software development project in order for the united work force to perform effectively. Therefore for the successful project, it is necessary to set management objectives to manage plan, and confirm if they are achieved. This method is considered to be effective, but actually planning a software development schedule and following-up the achievement of the management objectives at each step are not easy. Because since all the work for software development is performed in human brain, other people can not measure how much each worker exanimate each work. Therefore it is necessary to secure the time for measuring the depth of a work's examination by project manager's talk with the worker in charge, through automating as many parts as possible by means of mechanizing. It is difficult to make a software development plan itself, because predicting the necessary work amount and risks that the project involves is difficult in software development. Therefore, the authors are developing an automatic schedule planning system for software development so that the project manager can manage the entire project and the work load of the manager is reduced. This paper proposes a method to create automatically such a successful schedule plan that a project, which was behind schedule, will be completed on schedule by means of 'crashing'. And the paper proves that the method is effective in software project management through an example of system.

***Keywords*—** Crashing, software development Plan, Countermeasures, Process Delay

Daisuke Kinoshita, Shibaura Institute of Technology Graduate School of Engineering, 3-7-5 Toyosu, Koutou-ku, Tokyo, Japan,
m707103@shibaura-it.ac.jp
Rihito Yaegashi, Shibaura Institute of Technology Center for Information Science , 3-7-5 Toyosu, Koutou-ku, Tokyo, Japan
rihito@sic.shibaura-it.ac.jp
Kazuhiro Uenosono, Shibaura Institute of Technology Graduate School of Engineering, 3-7-5 Toyosu, Koutou-ku, Tokyo, Japan
k-uenosono@komiya.ise.shibaura-it.ac.jp
Hiroaki Hashiura, Shibaura Institute of Technology Graduate School of Engineering, 3-7-5 Toyosu, Koutou-ku, Tokyo, Japan
m705107@shibaura-it.ac.jp
Hiroki Uchikawa, Shibaura Institute of Technology Graduate School of Engineering, 3-7-5 Toyosu, Koutou-ku, Tokyo, Japan
m107014@shibaura-it.ac.jp
Seiichi Komiya, Shibaura Institute of Technology Graduate School of Engineering, 3-7-5 Toyosu, Koutou-ku, Tokyo, Japan
skomiya@shibaura-it.ac.jp

I. Introduction

Software project planning problems are formalized as follows. It is easy to understand the following-mentioned definition if you paraphrase a job into a project, and a machine into a worker.

The There are **n** jobs (***projects***) J_j (j=1, 2,…**n**) processed by **m** machines (***workers***) M_i (i=1, 2,…**m**). Each jobs (***projects***) J_j consists of n_j works A_u

$$Au\ (\ @\ =\ N_{j-1}+1, N_{j-1}+2,...,\ N_j; N_0; N_j\ =\ \sum_{k=1}^{j} n_k\ @)$$

, and some things which must be processed in ascending order of u are contained in Work A_u. Moreover, some works which requires some special machines (***workers***) are included in Work A_u. Each work may be simultaneously processed by two or more machines (***workers***), and each machine (***worker***) may simultaneously process two or more works. Each job (project) may be simultaneously processed by two or more machines (***workers***), and each machine (***worker***) may simultaneously process two or more job (***project***). The time required by each work varies according to the machine (***workers***) allocated to the work. The reliability of the intermediate products generated by each work varies depending on the machine (***worker***) allocated to the work. In such a situation, determine the schedules of all the works and each machine allocated to each work, so as to satisfy one or the combination of the following conditions:

(1)*The amount of a maximum completion time of each work is a minimum.*
(2)*The amount of cost is a minimum.*
(3)*Reliability of the software system to be generated is a maximum.*

This problem differs from ***JSP*** (***Job-shop Scheduling Problems***) in the following points:

(a)Any works do not require a special machine (***worker***) in ***JSP***.
(b)The processing order of all the works is given from the first in ***JSP***.
(c)No work is simultaneously processed by two or more machines (***workers***), and no machine can simultaneously process two or more work in ***JSP***.

(d)No job (***project***) is simultaneously processed by two or more machines (***workers***), and no machine (***worker***) can simultaneously process two or more jobs (***projects***) in ***JSP***.
(e)***JSP*** has only the above-mentioned ***(1)*** as an evaluation function of a schedule. However, the schedule planning problem for software project has not only the above-mentioned ***(1)***, but also the above-mentioned ***(2)*** or the above-mentioned ***(3)***, and the combination of ***(1)***, ***(2)*** and ***(3)*** probably. Therefore, the evaluation function of the schedule-planning problem for software project is more variegated and more complicated than it of ***JSP***.

It is human beings that conduct software development. However, it does not mean that any human beings can perform these works. In many cases, specific skills or qualifications are required. Since there are few people of such ability, a capable worker is sought after and requested to participate simultaneously at several projects in the situation where there are many projects parallel. This kind of prob1em never arose before the appearance of know1edge-intensive industry such as software deve1opment. It is impossible to organize a software development project unless there are a 1arge number of workers with high 1eve1 technica1 know1edge. To cope with this prob1em, the capab1e personne1 necessary (***human resource***) for each work and avai1ab1e period (***time with in the worker's schedu1e which is avai1able***) of the capab1e personne1 necessary must be taken into consideration when setting up a schedu1e for software development. The same holds true for non-human resources such as computers necessary for the project.

Therefore, the conc1usion is as follows.

Setting up a schedule for software deve1opment depends upon allocation conditions of human and non-human resources necessary for each work (this is ca11ed "the constraints concerning the resources allocation conditions") and avai1able period of human and non-human resources to meet the conditions (this is ca1led "the constraint concerning the avai1able period of each resource").

The authors have proved through showing a system execution example in [1] that the framework to develop a schedule planning system which was constructed using a genetic algorithm is effective in planning a schedule for software development. But the system proposed in [1] was improved so that several workers can be assigned to one process, as this system could not allocate several workers to one process. This paper proposes a system framework that several workers can be allocated to one process, and proves that the proposed framework is effective in developing a schedule planning system through creating automatically a schedule plan as a countermeasure against process delay by means of 'crashing' that is a method for duration compression.

The objectives of this paper are not to discuss the technology of GA (Genetic Algorithm), but to deal with and practically solve the problems peculiar to software project management.

The organization of this paper is as follows. Section2 clarifies the constraints that software schedule planning problems include and how to represent them. In section3 first clarifies the reasons for the introduction of GA into the schedule planning system in order to solve this problem, and proposes that the framework of a system to be able to allocate several workers to one process, as only one workers allocated one process in the former system. Section4 proves that the system framework is effective in developing software development schedule planning system through showing an execution example of the system to create automatically a countermeasure plan against process delay by means of 'crashing'. Section5 presents the conclusion of this paper.

II. RELATED WORK

Various meta-models in **PMDB** [6], **Design- Net** [4], **Kyoto DB** [5], and **PROMX** [7] have been already proposed to represent work structure of a software development project. However, **PMDB**, **Design-Net**, and **Kyoto DB** are not appropriate for generating schedules which take into account the constraints on resource allocation conditions and the available period of each resource because they clarify neither the relationship between the work in software development and the resources to use for executing the work, nor the constraints relating to the resource allocations conditions and the available period of each resources. (Although PMDB has Person as an entity, it does not deal with the constraints relating to the resource allocation conditions and the available period of the each resource). Therefore, these models are insufficient for presenting work structure of software development projects in order to develop software project management system.

We will perform comparison of CCPM (Critical Chain Project Management), which has become the center of attention recently with our approach. At first we explain TOC (Theory of Constraints) to argue about CCPM. TOC is the management method which by paying one's attention to the weakest portion (called constraint conditions by TOC) in the processes of the work in a company, and strengthening and improving that intensively achieve the greatest success with the minimum efforts.

Based on such idea of TOC, project management method, which attains optimization of the whole project, is CCPM. By using Critical Chain in place of conventional Critical Path, excluding the extra part of the time required to do a process for the safety security included in estimate of the process units, instead of shortening the period of each process (by adopting the estimate of time required to do a process with success of 50% probability), arrange margin time called project buffer over the last process in Critical Path, and condense time required to do a process for safety security and manage it. In addition, we insert joining buffer margin time between the work of Critical Chain and the joining work of Critical Chain to protect Critical Chain from the delay of Process of joining critical path. And by thinking about delivery date, cost and constraints of the resources the plan of project schedule is made. Project manager does not manage progress of each process after doing it in this way and grasps progress of the whole project by ratio of consumption of buffer.

Above is the outline of CCPM.

- The differences of CCPM and our approach are as follows. The views of time required to do a process estimate differ. In CCPM, by adopting time required to do a process estimate with success of 50% probability for time required to do a process estimate of each process and with joining buffer and project buffer, reduce danger of process delay by the estimate error. By our approach, we do averaging of the extra part of the time required to do a process by joining buffer and project buffer and divide it for each process.
- The views of progress management differ. In CCPM, progress of each process is not managed, but we manage progress of the whole project by ratio of consumption of buffer. For this reason, although process delays in the whole project can be grasped, it is not suitable for progress grasp of the processes, which are not on Critical Chain. In our approach, we manage progress for every process. For this reason, progress grasp is possible for all the processes, irrespective of the fact if they are or not on the Critical Path.
- When process delays are revealed, re-planning of the schedule proposal as a measure is not easy in CCPM, but in our approach as already shown in the documents, the tool which we developed can perform dynamically re-planning of the plan proposal in which process delays recovery is possible.

III. The Constraints Included Software Project Schedules Planning Problems and a Method for Representing the Constraints

This section discusses the constraints which software project schedule planning problems include, and how to represent them.

(1) Constraints concerning the execution order of activities

There are some intermediate software products to determine the order in which the works of software development are conducted. One example of the constraints existing between software development works and intermediate software products is that an intermediate software product 'α', which is necessary for execution of work 'b', must be obtained before work 'b' is executed in order for work 'b' to be conducted. This constraint is called 'a pre-condition'.

Another constraint is that an intermediate software product 'β' to be produced by execution of work 'b', must be obtained after execution of work 'b'. This constraint is called `a post-condition`.

The execution order of the process 'a', 'b', and 'c' are determined depending on the intermediate software products 'α' and 'β'. Such constraints are called 'ordering constraints'. In the system, `a pre-condition' and `a post-condition' of each process are put in a database in the form as shown in Table 1. The process generated on the basis of the ordering constraints defined by Table I is as Fig. 2.

(2) Constraints concerning the conditions for resource allocation

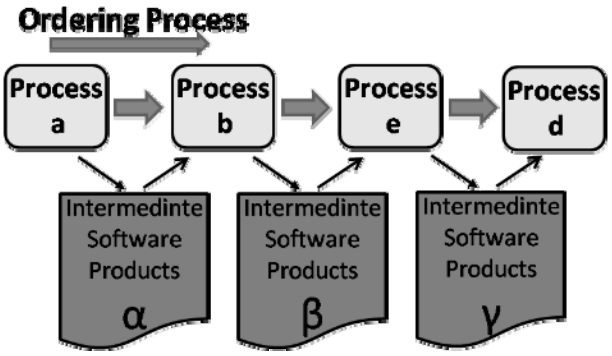


Fig. 1 Ordering Process by Intermediate Software Product

Each work of software development has constraints that only the resources (i.e. workers) with skills, qualifications, or

TABLE I
`A PRE-CONDITION' AND `A POST-CONDITION' OF EACH PROCESS

Process	Pre-conditions (required Intermediate software Product	Post-conditions (obtained Intermediate software Product)
SA	a	b
SD1	b	c
SD2	b	d
DR	c,d	e
DD1	e	f
CT1	f	g
DD2	e	h
PE	h	I
CT2	i	j
DT	e	k
IT	g,j,k	l

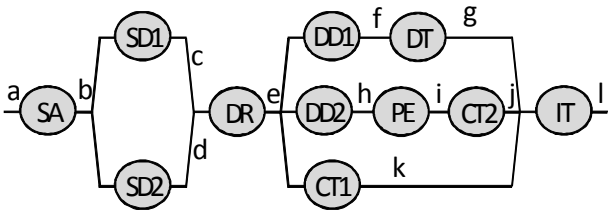


Fig. 2 The Process generated on the basis of constraints defined by Table I

functions, which are required for execution of the work, can be allocated to the work (i.e. 'constraints concerning the conditions for resource allocation'). For example, a worker with each capability needs to take charge of processes, such as a program development language, and system test, debugging. Therefore, a schedule for software development is determined by the constraints of each work in relation to the allocation of human and non-human resources.

By the way, the following constraints are classified into 'Constraints concerning the conditions for resource allocation': "A boss should lead his junior partner who troubled a customer not to go to the same customer." Because the junior partner does not have such a qualification to be allocated to the resources in a wide sense.

(3)Constraints concerning Available Period of Each Resource

Even if each work of software development satisfies conditions which can be allocated, it has constraints that resource can be allocated only in the period which can be allocated. The result of this allocation is described in the method explained in [1].

(4)Constraints concerning each resource's Limitations

The concept of capacity is introduced and expressed as an attribute of each resource to express resource capability limitations. A resource's capacity is defined in terms of the upper limit value of each resource's working rate (expressed in percentage terms). The working rate is obtained by dividing the amount of working hours allocated to a resource per day (when one resource is allocated to several works, the amount of working hours for the combined works is calculated) by the available working hours of that resource and that result is multiplied by 100. The working rate should be calculated in advance and the working rate upper limit value is defined as the resource capability limitation. For example, suppose worker Pl is allocated to work A (two days necessary for execution) and work B (two days necessary for execution) for five days, the working rate of worker P1 for the week is 80%. In this case, if the working rate upper limit value of this worker is more than 80%, the above assignments are possible. However if the value is below 80%, it is not possible to assign the worker on the above assignments. The working rate can be utilized as a scale for evaluating the worker's workload and for detecting that the worker is over loaded. This concept also applies to non-human resources. It is considered that the capacity in general (working rate upper limit value) differs according to the rank of the resource.

IV. The Reasons for the Introduction of GA into the Software Schedule Planning System and How to Specify the Number of the Workers to be Assigned to Each Process

A. The reason for applying GA to schedule planning problems for software development

The fundamental mechanism of the computation model of GA is 'Generate & Test'. That is, the candidates of solutions are generated one after another, and the solutions are selected from among them according to the degree of satisfaction of a solution. Therefore, the number of combination increases, as GA generates the combination of chromosomes one after another. Consequently computation time increases exponentially. That is, there is a fault that computation time becomes huge when the number of combination increases. However, even if the number of constraints or the number of the evaluation functions of a solution increases, there is an advantage that computation time hardly increases. By the way, the schedule planning problem for software development can be regarded as a combination optimization problem with the following characters. That is, although the number of the combination of the work items and resources becomes huge, the number of the combination to be examined as candidates of solutions is not so many, as the problem has many constraints, for example, constraints concerning the execution order of activities, constraints concerning the conditions for resource assignment, and constraints concerning available period of each resource and so on. Therefore, the schedule planning problem for software development does not require so many computation time. Moreover, GA has the advantage that the persons who have domain knowledge can easily obtain approximate values of solutions, as the evaluation function of solutions is given in the form of the constraints in GA, even if they have no knowledge of a solution. Moreover, GA has the advantage that it is easy to modify or change a model, even if there is the omission of constraints or errors, or strategy change. For reasons of these advantages, we adopted GA into schedule planning problem for software development.

B. How to specify the number of the workers to be assigned to each process

We evaluated capability of each worker according to the four-grade system in terms of excellent, good, poor, and impossible in this research.

Excellent-grade means that the worker can conduct the work for oneself quickly and teach another worker or other workers how to conduct the work while conducting his/her works. Good-grade means that the worker can manage to conduct the work for oneself but cannot teach another worker how to conduct the work while conducting his/her works. Poor-grade means that the worker cannot conduct the work for oneself, but cannot conduct the work according to another worker's instruction. Impossible-grade means that the worker cannot conduct the work even if the work receives other worker's instruction.

Usually the number of works to be allocated to each process is two or more. Suppose that the allocate patterns of the works to be allocated to each process are like Table II.

TABLE II
EXAMPLE OF HUMAN ASSIGNED PATTERN (IN CASE OF CT1)

	Pattern 1	Pattern 2	Pattern 3
A(Excellent)	1	1	
B(Good)			2
C(poor)		1	
D(Impossible)			
Necessary days	3	2	2

An example is shown in Table II and explained. There are three allocation patterns of the workers to be allocated to this process. The pattern 1 shows that this process requires the 3 days to conduct this process, if a worker who has the skill of < excellent> level, which is required to conduct this process, can alone conducts this process. The pattern 2 shows that this process requires the 2 days to conduct this process, if a worker who has the skill of < excellent> level and a worker who has the skill of <poor> level can conduct this process. The pattern 3 shows that this process requires the 2 days to conduct this process, if two workers who have the skill of<good> can level conduct this process. The number of the workers is specified on the basis of the pattern of worker's allocation as stated above. GA evaluates all the allocation patterns of the workers to be allocated to each process in order to allocate these workers to each process.

A candidate of the execution order of activities and allocation patterns of the work are shown Fig.3. The candidate who is created by using GA is organized at the first layer and the second layer. Fig.3 shows that the first layer expresses the execution order of activities and the second layer expresses the allocation patterns of the work. For example, Fig.3 shows that allocation patterns of the work to CT1 have three patterns { Pattern-1 Pattern-2 Pattern-3 }. And the third layer expresses

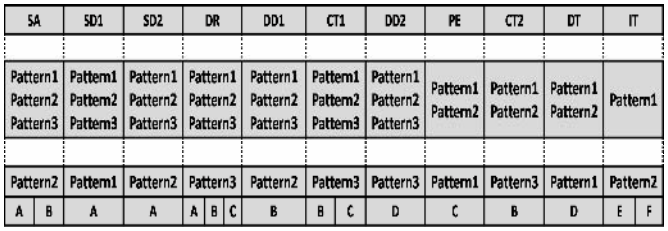


Fig. 3 How to specify the number of the workers to be assigned to each process

of a system to create automatically a schedule plan as countermeasures against process delay by means of "crashing". "crashing" is a method for duration compression[8].

Outline of Project Used as a Hypothetical Example

[Hypothetical example]

Suppose there is a project to develop a demonstration system to present in an exhibition that is to behold from August 1. This project is to be organized on July 2 and the system should be completed by July 31. The necessary activities for this system development are basic design, outline design, outline design review, detailed design, coding, individual testing, and comprehensive testing. The constraints to be taken into consideration for planning the project are listed below. Since the demonstration system requires high performance, performance estimation should be done on part of the detailed design and the result be reflected in the coding work. Performance estimation requires performance analysis technology. The hardware that is to be utilized in the exhibition is a new machine developed for this exhibition and can be utilized from July 24 at the earliest. This machine should be transported to the exhibition hall by July 31. Due to the constraint with regard to the period that the machines will be used in the exhibition, it has been decided that a test support tool will be developed for efficient testing. Under the conditions described above, a schedule was planned and the project began.

Role for each project human resource is listed below

- The Project Manager is in charge of project planning, progress management, and giving approval.
- The Design Engineer is generally responsible for designing

TABLE III
THE VALUE OF HUMAN RESOURCES ATTRIBUTES

Name	Skill	Available time	Cost / h
X	Manager	Any	3000
A	OOA, C	Any	2800
B	SA, C	Any	2300
C	JSD	7/15-7/18 8/1-8/5	3000
D	Performance	7/2-7/20 7/28	3500
E	Analysis	Any	3000
F	SA, C	Any	3000
G	SA, C	7/13-	3500
H	SA	-7/18	3200

TABLE IV
THE VALUE OF NON-HUMAN RESOURCES ATTRIBUTES

Name	Available Time
Performance Analysis Tool	7/10-7/18 8/1-8/5
Test Support Tool	7/23-
Machine for Development	Any
Machine for Exhibition	7/26-7/30

g. 4 Activity diagram in this project

of the July 11. DR is located on a critical path as you can understand by PDM (Precedence Diagramming Method) shown in Fig.6.

By the way, process delay of DR means the process delay of a whole project, as the process delay of a process on the critical path means the delay of a whole project. Fig.5 shows that one day's delay eventually changes a delay of 15days, due to constraints concerning available period of the Worker C. Because the worker C cannot conduct the work of PE(Performance Evaluation) until August 1 due to the constraints concerning available period of the worker C who takes charge of the work of PE (See Table.III).

Therefore, a system creates automatically a schedule plan to eliminate process delay by adding newly a resource or several resources (i.e. worker(s) in this case) to the process after July 12. It is called 'crashing' to eliminate process delay by adding newly a resource or several resources to the process on a critical path [9]. Fig.6 shows that process DD2, PE, CT2, and IT are located on a critical path, and IT (Integration Test) must be conducted on schedule due to the constraints concerning available period (from July 26 to July 30) of resources. Furthermore, IT must be assigned a worker with a special skill, as IT requires special skill. Therefore, it is required to create a schedule plan as countermeasures against process delay that completes system implementation by July 31, through adding a worker or some workers to some of DD2, PE, CT2, and IT newly. This mechanism is "crashing".

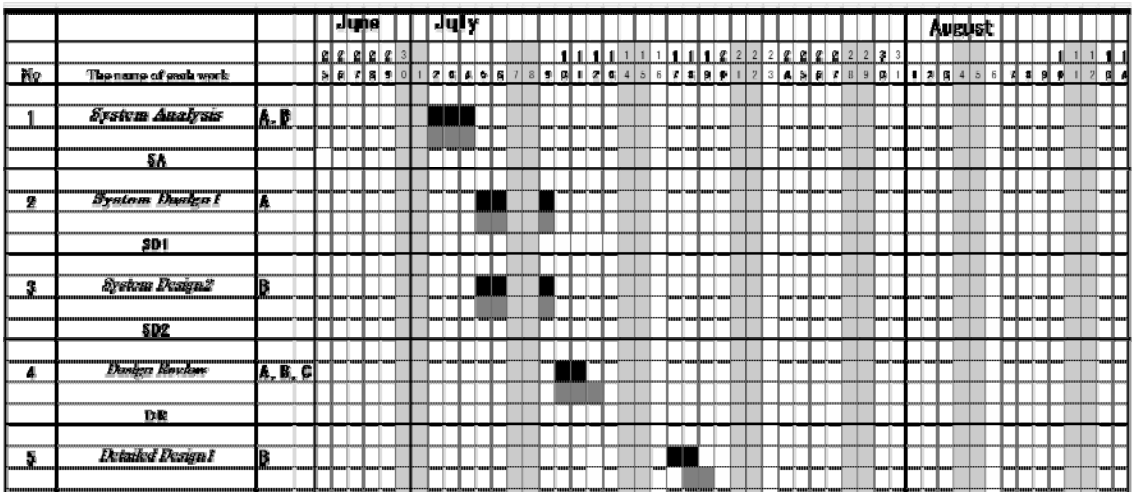
A system that we developed created two schedules plans as countermeasures against process delay that eliminated process delay (See Fig. 7, Fig. 8), when our tool adapted to the example project. The countermeasure schedule plan 1 adds an excess worker G to DD2 (Detail Design2), and two workers of B and G takes charge of DD2, Although Worker B is going to complete the process in three days in an original plan, in the countermeasure schedule plan 1, two workers of B and G take charge of DD2, and DD2 is going to be completed in two days. A countermeasure plan 1 of Fig.7 shows that system development can be finished by July 31. A countermeasure plan 2 of Fig.8 adds an excess worker H to DD2, two workers of B and H take charge of DD2, and DD2 is going to be completed in two days.

Notice that these solutions are satisfies the following constrains:

The excess workers who can assign are only G and H. The periods, which can be assigned to worker, G and H are since July 13, and since July 18. Moreover, the worker H can take charge of system analyst, and use programming language C. Worker G can take charge of the job of system analyst.

In an example project, our system creates only a plan as countermeasures that an excess worker was added to DD2. This is the reason for the constraints that human/non-human resources to be assign to PE (Performance Evaluation) have. This is because a special skill and a special tool are required in order to conduct PE (Performance Evaluation).

This is because process delay must be eliminated before PE is carried out, as neither of the resources can be used until July 18.



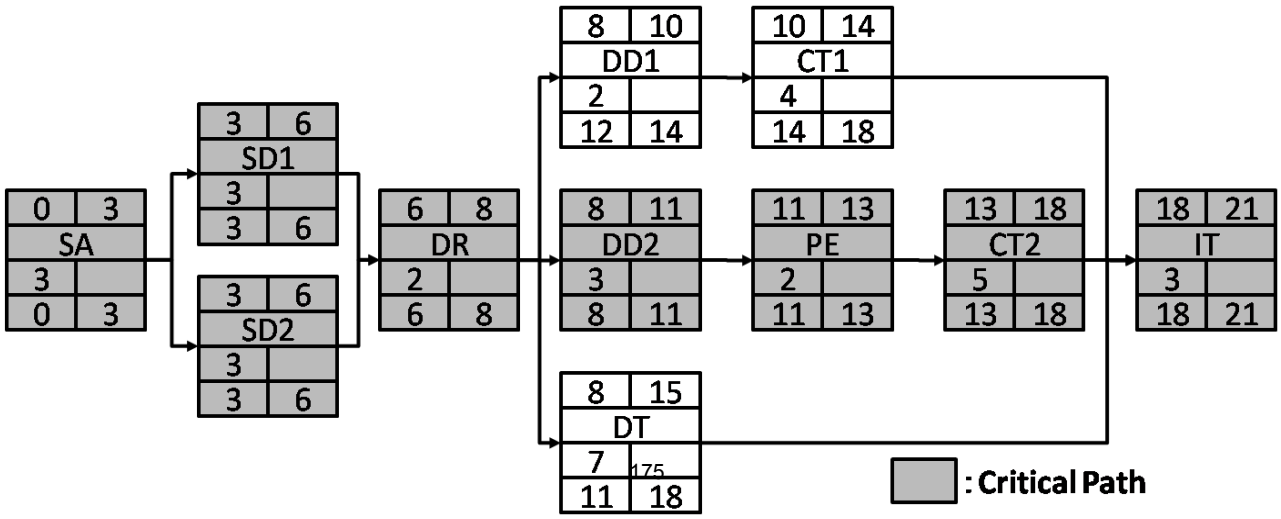


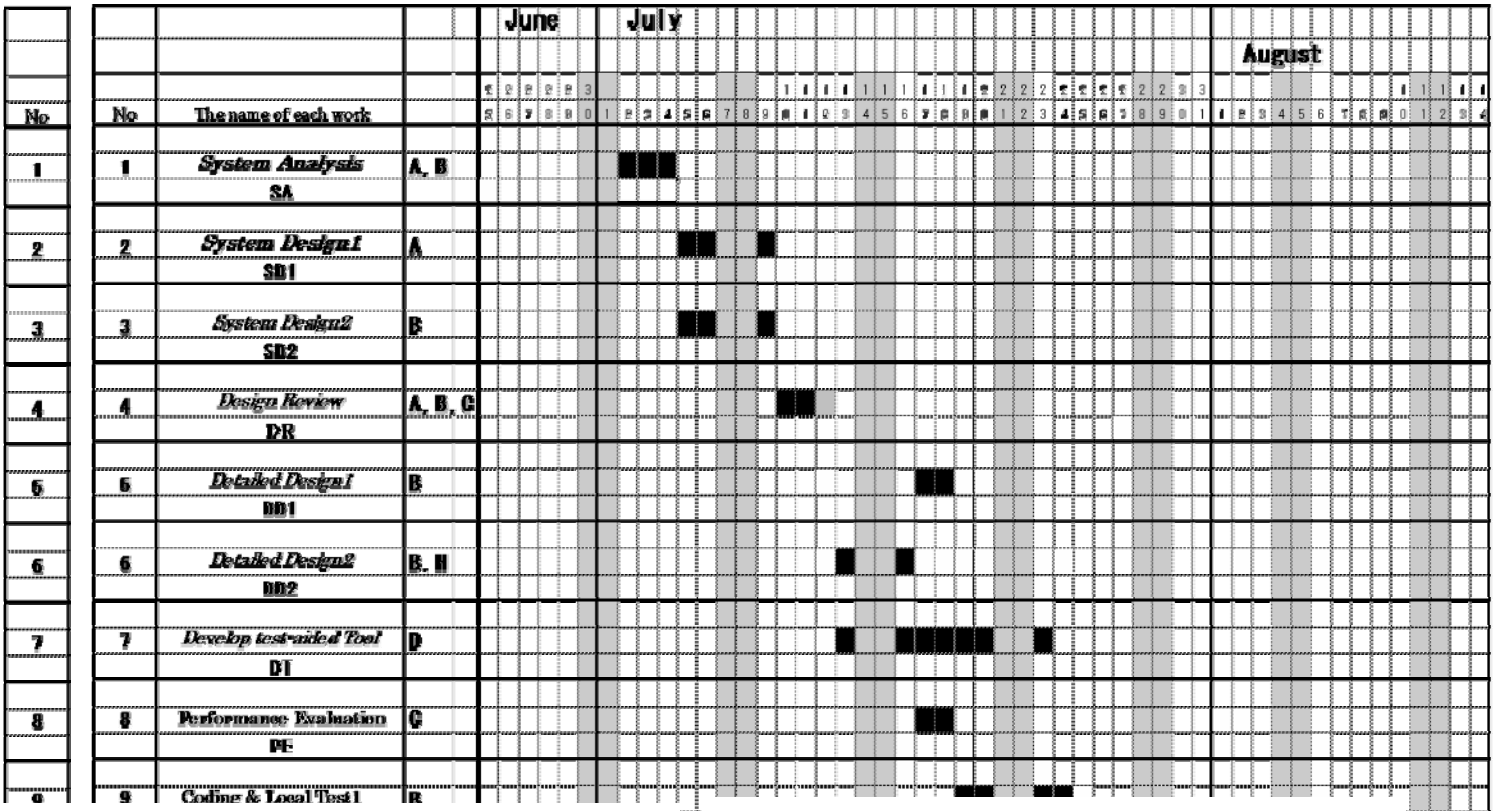
Fig. 6 Measurement of the critical path base on PDM

The countermeasure plan (Fig.9) was adopted in the example project.

This is because a strategy 'cost minimum' was adopted for planning. If other strategies are adopted, other schedule plans will be create. This means that other strategies can be set up by changing the evaluation function of GA.

This system presents a software project plan in terms of 2-layered chromosome. In old system, the upper layer presents the execution order of processes (or works), the lower layer

n workers one time of scheduling to a work by using one of the worker assignment patterns, the workers to be assigned by the new system are n times as much as the workers to be assigned by the old system, If you assign the same numbers of workers by using the old system, you must repeat the assignment n times. That is, the more the scale of a project is larger, the more the effect is larger. Therefore, we think you can understand that this improvement largely contributes to reducing the troublesome of software project management.



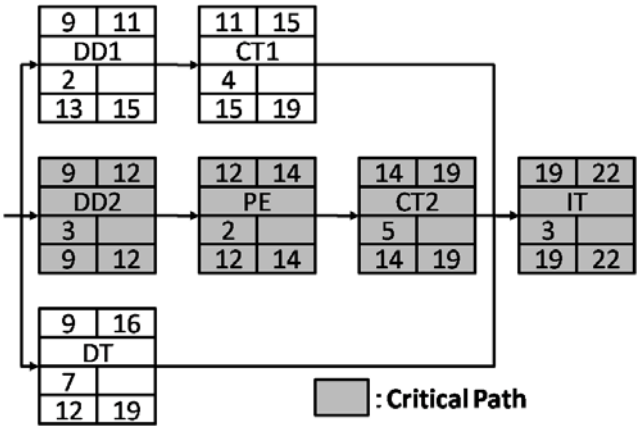


Fig. 9 Measurement of the critical path base on PDM (after delay)

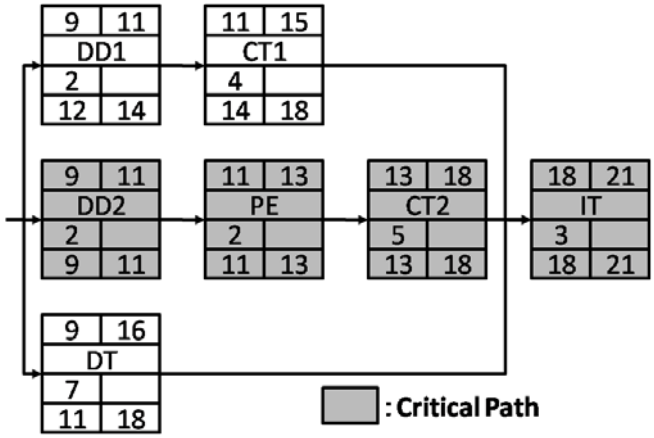


Fig. 10 Measurement of the critical path base on PDM (Re-plan proposal)

VI. Conclusion

This paper clarified the constraints to be taken into consideration, and described how to represent these constraints, in order to solving a schedule planning problem for software developments.

Moreover, we clarified the reasons for the introduction of GA into schedule planning problem for software development in order to solve this problem, and proposed the system framework to be able to assign several workers to one process. Finally we proved that the system framework is effective in developing software development schedule planning system through showing an execution example of the system to create automatically a schedule plan as countermeasures against process delay by means of "crashing" (a method of duration compression), and proved that it also is effective in re-planning on the way of software development, as the system is implemented by using GA is easy to modify the model.

Acknowledgment

The authors would like to thank all members of Software Engineering and Knowledge Engineering Laboratory (Komiya Laboratory), Shibaura Institute of Technology for their help in developing the prototype system.

References

[1] S.Komiya N.Sawabe and A.Hazeyama, Constraints-Based Schedule Planning for Software Development, *IEICE Trans. Inf. & Syst*., vol. J79-D-I, no9, 1996, pp544-557.

[2] S.Komiya A.Hazeyama, A Meta-Model of Work Structure of Software Project and a Framework for Software Project Management System, IEICE *Trans INF & SYST*, vol.E81-D, No12, 1998, pp1415-1428.

[3] A.Hazeyama and S.Komiya, Workload Management Facilities for Software Project Management, *IEICE Trans Inf & Syst*., vol. E81-D, No12, 1998, pp1404-1414 .

[4] Penedo M.H. and Stuckle, E.D., PMDB - A Project Master Database For Software Engineering Environments, *8th International Conference on Software Engineering*, 1985, pp.150-157

[5] Liu, L and Horowitz, E., Object Database Support for a software Project Management Environment, *ACM SIGSOFT Software Engineering Notes* Vol.13, No5, 1988, pp.85-96

[6] Matsumoto Y and Ajisaka T, A Data Model in the Software Project Database KyotoDB, *JSSST Advances in Software Engineering Environments, (the International Conference on Software Engineering*, 1985, pp.150-157

[7] Sato, H., Project Management Expert System, Proc.ACM CSC 87, 1987

[8] Project Management Institute, *A Guide to the Project Management Body of Knowledge (PMBOK Guide) 3rd Edition*, Project Management Institute, 2004