Automatic Creation of Countermeasure Plans against Process Delay: Creation of Countermeasures based on Holiday Work

Kazuki MORI†, Daisuke KINOSHITA††, and Seiichi KOMIYA†
†Graduate School of Engineering and Science, Shibaura Institute of Technology, ††Hitachi, Ltd
†3-9-14 Shibaura, Minato-ku, Tokyo, 108-8548, ††6-27-18 Minamioi, Shinagawa-ku, Tokyo, 140-8572
Japan
ma11157@shibaura-it.ac.jp

Abstract: - The authors have been developing the automatic scheduling systems for software development project, which deliver scheduling and worker assignment. Their earlier works were designed to draw on crashing and fast tracking to produce countermeasure plans against process delay. However, process delay can be recovered not only by crashing or fast tracking but also by using holiday works. Their earlier works did not take holidays into the considerations as a constraint, and supposed that the holiday works were unassignable because the holidays were considered to be an unchangeable object (from application perspective) and were treated implicitly. In this paper, the authors consider the holidays as a constraint against worker assignable days, and treat explicitly handle them. Therefore this paper proposes a mechanism for producing countermeasure plans by using holiday works, and proves that the proposed mechanism is effective in producing countermeasure plans stated above.

Key-Words: - Schedule Planning for Software Development, Countermeasure Plans against Process Delay, GA, Holiday Work

1 Introduction

In many cases, software development projects tend to cause some project process delay due to underestimated efforts or overestimated workers' skill. Consequently, the Project Managers will need to develop some schedule for the countermeasures against such process delay by considering various conditions including workers' schedule and/or skills. There are three (3) possible types of countermeasure plans against process delay; the countermeasure based on crashing, on holiday works, and on fast tracking. It seems that the best way to make a decision on the countermeasure plan is to compare these three types of the plan and then select the would-be best plan. (This paper does not consider "reduction," which means a part of the system will be abandoned, as our system does not support it).

Therefore, this paper proposes a method to create a countermeasure plan using holiday works and demonstrates its effectiveness since "countermeasure plan based on crashing" and "countermeasure plan based on fast tracking" are already proposed in References [3] and [6] respectively.

This paper consists of the following sections. Chapter 2 defines the term "holiday works" as used in this paper, and describes holiday categories stipulated by Labor Standards Act. Then it outlines "compensation day," which should be considered in return for holiday work, and explains how to apply for the compensation day. Chapter 3 introduces the constraints of software development planning problems, which will serve as the assumptions of software development planning. Then it shows that the software development planning problems equate to solve the Combinational Optimization problems of the resources that meet the constraints, and it also indicates that Genetic Algorithm (GA) is used for solving the Combinational Optimization problems. Chapter 4 discusses how to give the number of workers assigned to each phase. Chapter 5 shows the difference between conventional and proposed worker assignment mechanisms. Chapter 6 verifies the effectiveness of the mechanism as proposed in this paper through a simple example. Chapter 7 compares this study and relevant studies. Section 8 describes the conclusion.

2 Statutory Definitions for Holidays and the Concept of "Holiday Works" in this System

By definition of labor laws, a holiday means the day on which a worker is exempt from his/her obligation to work under the labor contract [20].Clause 35 of the Labor Standards Act provides two types of the holiday; that is, statutory holiday
(the worker shall be given at the minimum of one (1) holiday per a week) and non-statutory holiday (which is to be given outside the laws). In this study, working on these holidays are referred to as "holiday works." Clause 36 of the Labor Standards Act stipulates that a compensation day shall be given only when a worker works on a statutory holiday, but, in general, the worker who works on non-statutory holiday is often entitled to seek for a compensation day under the provisions of the labor union and/or company rules. In addition to that, the employer is not required to pay a premium for the worker who works on the non-statutory holiday unless he/she works more than eight (8) hours per day or forty (40) hours per week. Therefore, this research assumes that the worker will have one (1) compensation day to the greatest extent possible when he/she performed one (1) holiday work, aiming to control project cost increases. When the worker was not able to have a compensation day, the extra cost should be calculated by applying percentage rate of premium to his/her hourly pay for the holiday works.

As the companies usually manage the cost by project and it is convenient to have a clear correspondence between the holiday works and the compensation days for the purpose of labor management, it is preferable to have the compensation day within the project period in which the holiday works were done.

3 Constraints on Software Development Planning Problems

This research regards the requirements of any software development plan as constraints. Detailed description of software development planning problems is given below.

(C1) Constraints on operational sequence

Actual operational sequence of the software development processes depend on intermediate products. For example, Process b of Figure 1 can be explained as follow.

Operation of Process b requires that the product of Process a (Intermediate product) α is created in prior to the start of Process b. This condition is referred to as "the precondition of Process b." And, the product of Operation of Process c requires that the product of Process b (Intermediate Product) β should be created in prior to the start of Process c. This condition is referred to as "the post-condition of Process b." Thus the operational sequence of Process a, b, c is determined by Intermediate Product α, β. These constraints are referred to as "constraints on operational sequence."

(C2) Constraints on resource assignment conditions

Each task of the software development should be assigned with the human resources (worker) and/or non-human resources (e.g. machine environment) that have necessary skills, qualifications and/or capabilities for that particular task. These constraints are referred to as "constraints on resource assignment conditions." For example, programming language, system testing, debugging and other processes require the workers who have respective competencies. As a result, the software development work schedule depends on the constraints on human and non-human resource assignment conditions of relevant tasks for the software development.

(C3) Constraints on resource assignable periods

Furthermore, each task of software development has another constraint that the qualified resources are available only for their assignable period (i.e. when they are not fully booked and available for such task). These constraints are referred to as "constraints on resource assignable periods."

(C4) Constraints on resource capacity limitation

This paper introduces the concept of “capacity” in order to represent respective resource capacity limitation, and present it as an attribute of the resources. More specifically, the capacity of given resource is represented by the upper limit of its working rate (in percentage). That is, the total working hours for a day as assigned to the resource (or, the total working hours for the set of tasks in the event that single resource is assigned to a set of tasks that should be performed in parallel) should be divided by daily workable hours for such resource and then the result should be multiplied with 100 to derive the working rate. The predefined upper limit for the working rate of each resource is referred to as the constraints on resource capacity limitation of given resource. By implementing these constraints, the working rate can be used as a scale for evaluating a worker’s workload and for checking if the worker is overloaded. The same concept is also applied to non-human resources. In general, capacity (upper limit of working rate) is likely to vary pursuant to the rank of such resource. Although
the upper limit of assignable working hours per day is set as eight \((8)\) hours, default working rate in this study is \(80\%\) with consideration to work breaks and other intermissions.

(C5) Constraints on worker assignable days

Conventional systems \([1-7]\) create the software development plan by assigning the works exclusively on weekday, which means the day other than the holiday and is allowed to assign a work to a worker (typically referred to as weekday). This paper assumes that the works will be performed according to this development plan, and when there is some process delay, such a delay should be recovered by assigning the workers, even on the holidays, for the works of certain user-specified process. During the course of this arrangement, holidays are regarded as one of the constraints on worker assignment and are handled, in principle, as "the day that is not subject to worker assignment." However, a limited number of processes as specified the user of this system should be handled as "the day that is subject to worker assignment." This arrangement enables certain user-specified processes to have mitigated constraints on worker assignment. These constraints are referred to as "constraints on worker assignable days."

When the combination of resources as assigned to each process of the project meet all of the constraints, such combination can be regarded as a candidate for software development plan. In other words, creating a software development plan can be compared with solving a Combinational Optimization problem with many constraints. The authors employ Genetic Algorithm (GA) as a mechanism to solve such an optimization problem. Underlying mechanism of GA calculation model is characterized as "Generate & Test." Consequently, GA model generates potential solutions one after another, tests them whether they can meet the constraints, and obtains the candidate solutions that cleared the test. Then the model finally selects the solution with the highest evaluation value. References \([1, 2, 4]\) have already explained the reasons for applying GA to optimization problems, and have demonstrated that the GA is effective for software development planning problems.

Meanwhile, this paper does not consider non-human resources so that we can develop simplified explanation in the subsequent chapters.

4 How to Determine the Number of Workers to Be Assigned to Each Process

This research evaluates worker's competencies in four levels of "Excellent," "Good," "Poor," and "Impossible." Those who have "Excellent" level of capability can work quickly on his/her own and give guidance to "Poor" level workers while doing his/her job. Those who are at the "Good" level can work on his/her own but are unable to lead others while working. "Poor" level workers can work if he/she is under the supervision of other worker. "Impossible" means whose who are not capable of working even if he/she is under supervision of other worker. Usually, a number of workers are assigned to each task. Consider that the worker assignment pattern for a certain process is as shown in Table 1. In the case of Table 1, there are three patterns in the number of the workers to be assigned by the worker assignment for this process. Pattern 1 indicates that it will take three days for those who have "Excellent" level of capability (required for this process) to complete the task. Pattern 2 shows that it will take two days for an "Excellent" level worker and a "Poor" level worker to complete the task. Pattern 3 requires two days for two "Good" level workers to complete the task. On the basis of these worker assignment patterns, the number of workers required for each process can be specified. This system assigns workers by executing and evaluating all of the specified worker assignment patterns.

Worker assignment patterns assume that the Project Manager estimates the workload by process in advance and input them to the system in the format of Table 1. This paper assumes, in the subsequent pages, that these worker assignment patterns are already loaded to the system.

5 Differences between Existing Mechanism and the Newly Proposed Worker Assignment Mechanism

This system employs the tiered coding approach in which GA chromosomes are segregated into two tiers of the upper and the lower tiers as shown in Figure 2, aiming to represent software development planning problems. The upper tier shows each working process by individual gene, and operational
Figure 2 The structure of two-tiered GA model

sequence is shown by chromosomes which represent lines of genes and the operational sequence. In the Lower tier, each gene represents a worker assignment pattern, and individual gene corresponds to its counterpart that has the same gene locus (gene location on the chromosome) for the upper tier and the lower tier. For example, Figure 2 shows that Pattern 2 is adopted for development of test support tool (DT). The mechanism traditionally adopted for worker assignment can be briefly described in the following procedures.

1. Generate candidate process lines and evaluate whether they are meeting the constraint conditions (exclusively retain the candidate that meets the constraints).
2. Generate candidate worker assignment pattern lines and evaluate whether they are meeting the constraint conditions (exclusively retains the candidates that meet the constraints).
3. Use the PDM method to automatically calculate the margin days (i.e. the day in which you do not have to start working immediately after completing the prior work) and the critical path.
4. Generate the schedule patterns (with different margin days) for each process.
5. Generate candidate schedule pattern lines and evaluate whether they are meeting the constraint conditions (exclusively retain the candidates that meet the constraints).
6. Assign the workers to the processes on non-critical paths as per worker assignment pattern and evaluate whether the processes meets the constraint conditions (exclusively retain the candidates that meet the constraints).

The difference between the conventional system and this system is that the concept as set forth in Reference [6] does not regard a holiday as a constraint on the worker assignment, rather, implicitly treated them (i.e., as an unchangeable object from the perspective of application). In contrast, the concept of "Generating countermeasure plan using the holiday works" regards all of the holidays as "constraints on worker assignable days," and, in principle, treats them explicitly as the days not available for worker assignment (i.e., modifiable objects from the perspective of the application). However, several user-specified processes are treated as open to the worker assignment. This arrangement allows the system to automatically generate the candidate countermeasure plans that assign the workers on the user-specified date. Then the system shortlists the candidate countermeasure plans by evaluating whether they meet the constraints on labor issues under two different conditions of "statutory holiday" and "non-statutory holiday."

6 An Example for Automatic Creation of Countermeasure Plan Using Holiday Works and its Assessment

This chapter gives a realistic example of a process delay that happened to a proposed development plan and verifies the effectiveness of our proposed approach by enabling automatic creation of countermeasure plan in the exemplar situation.

[Operation overview]
Consider a project to develop a demonstration system for a certain exhibition that will take place from August 3, 2010 as an example project. This project was organized on July 1, 2010 and is expected to complete the system by August 2, 2010. The system development
requires system architecture design, system design, system design review, detailed design, coding, unit test, and integration test. Constraints to be considered at proposed development planning are as follows. Since the demonstration system is required to be a high performance system, a performance evaluation is required on a section of the detailed design and the evaluation result must be reflected to the coding tasks. Conducting the performance evaluation requires the performance analytic technology. The hardware used in the exhibition is a newly developed machine that is now under development for presentation, and will be available as early as on July 26. Furthermore, it will be transported to the exhibition hall on August 3. Since the available period of this machine is under constraint as described above, a test support tool is developed to enable an efficient test. The project has started with development planning under such circumstances.

Workers for the example project should have the following attribute value respectively.

Project Manager: project management, progress management, approval activities.

Design Engineer: basic duties are designing and coding tasks.

Quality Assurance Engineer: basic duties are planning and executing the test.

Table 2 shows the necessary skills for each process. Workers are appointed to each role as follows: Worker X for Project Manager, six workers (Workers A, B, C, D, E, F) for Design Engineer, and three workers (Workers G, H, I) for Quality Assurance Engineer. These are the workers to be assigned to the processes. The attribute values of each human resource are shown in Table 3. In addition, the project uses a development machine, a production machine at the exhibition, a tool for performance evaluation, and a tool to be developed for efficient testing. The attribute values of non-human resources are shown in Table 4. The Activity Diagram of this example project is shown in Figure 3. The worker assignment pattern to each process is shown in Table 5 (assuming one worker assignment pattern for each process for simplification purpose). The working hours per day should be eight hours.

With consideration to the above mentioned attribute values and margin days, the project team established the proposed original development plan (upper tier of each process in Figure 4), and continued to work as along this development plan. Then the team found that they were one day behind the schedule at the time of July 12 due to a process delay in Process DR. So the expected end date of Process DR would be July 14 (mid-tier of each process in Figure 4). As a result, the most likely end date of the project would be August 8, which means significant overdue. The team established a countermeasure plan against the process delay by using our proposed approach. The countermeasure plan against the process delay is shown in the lower tier of each process in Figure 4.

Table 2 A list of the skills required for being assigned to each process.

<table>
<thead>
<tr>
<th>Process</th>
<th>Necessary Skills</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA (System Analysis)</td>
<td>OOSE, OMT, Cprogram</td>
</tr>
<tr>
<td>SD1 (System Design1)</td>
<td>OMT, TDD, Cprogram</td>
</tr>
<tr>
<td>SD2 (System Design2)</td>
<td>OMT, TDD, Cprogram</td>
</tr>
<tr>
<td>DR (Design Review)</td>
<td>OMT, Cprogram</td>
</tr>
<tr>
<td>D1 (Detailed Design1)</td>
<td>OMT, Cprogram</td>
</tr>
<tr>
<td>D2 (Detailed Design2)</td>
<td>OMT, Cprogram</td>
</tr>
<tr>
<td>DT1 (Develop-test-ended Tool)</td>
<td>Cprogram, AOP</td>
</tr>
<tr>
<td>PE (Performance Evaluation)</td>
<td>Cprogram</td>
</tr>
<tr>
<td>GT1 (Coding &amp; Unit Test1)</td>
<td>PA (Performance Analysis)</td>
</tr>
<tr>
<td>GT2 (Coding &amp; Unit Test2)</td>
<td>Cprogram</td>
</tr>
<tr>
<td>IT (Integration Test)</td>
<td>Cprogram, Jrlnk</td>
</tr>
</tbody>
</table>

Table 3 The attribute values of each human resource.

<table>
<thead>
<tr>
<th>Name</th>
<th>Skill</th>
<th>Skill Level</th>
<th>Available time</th>
<th>Cost ($/h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>OOSE, OMT, PA</td>
<td>AExcellent</td>
<td>7/21, 7/23~</td>
<td>4400</td>
</tr>
<tr>
<td>B</td>
<td>OMT, TDD, Cprogram</td>
<td>BGood</td>
<td>Any</td>
<td>2500</td>
</tr>
<tr>
<td>C</td>
<td>OMT, TDD, Cprogram</td>
<td>BGood</td>
<td>Any</td>
<td>3000</td>
</tr>
<tr>
<td>D</td>
<td>OMT, TDD, Cprogram</td>
<td>BGood</td>
<td>7/9, 7/20~</td>
<td>2500</td>
</tr>
<tr>
<td>E</td>
<td>Cprogram</td>
<td>CExcellent</td>
<td>Any</td>
<td>3500</td>
</tr>
<tr>
<td>F</td>
<td>OMT, TDD, Cprogram</td>
<td>BGood</td>
<td>7/9<del>7/16, 7/26</del></td>
<td>2500</td>
</tr>
<tr>
<td>G</td>
<td>Cprogram, Jrlnk</td>
<td>AExcellent</td>
<td>Any</td>
<td>4200</td>
</tr>
<tr>
<td>H</td>
<td>Cprogram, Jrlnk</td>
<td>BGood</td>
<td>Any</td>
<td>3500</td>
</tr>
<tr>
<td>I</td>
<td>Cprogram, Jrlnk</td>
<td>BGood</td>
<td>Any</td>
<td>3500</td>
</tr>
</tbody>
</table>

Table 4 The attribute values of each non-human resource.

<table>
<thead>
<tr>
<th>Name</th>
<th>Available time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Analysis Tool</td>
<td>Any</td>
</tr>
<tr>
<td>Machine for Development</td>
<td>Any</td>
</tr>
<tr>
<td>Machine for Exhibition</td>
<td>7/16~8/2</td>
</tr>
</tbody>
</table>

Table 5 The worker assignment pattern to each process.

<table>
<thead>
<tr>
<th>Process</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Necessary Days</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA (System Analysis)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>SD1 (System Design1)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>SD2 (System Design2)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>DR (Design Review)</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>D1 (Detailed Design1)</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>D2 (Detailed Design2)</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>DT1 (Develop-test-ended Tool)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>ET (Performance Evaluation)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>GT1 (Coding &amp; Unit Test1)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>GT2 (Coding &amp; Unit Test2)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>IT (Integration Test)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3 The activity diagram of an example project.
Our observation starts with a comparison between the proposed original development plan and the expected process delay. It should be noted that the original end date was August 2 under the proposed original development plan, but this end date was revised due to process delay to be August 8, which means there were six days of delay. This delay was caused by the fact that Worker A, who was assigned to process PE, was not available for the period of July 22 - 28 because of (C3), and resulting delay of one day in process DR caused total six days of process delay. Furthermore, calculating the project cost reveals that the cost will increase from Yen 1,256,000 for the proposed original development plan to Yen 1,868,000 after the delay, incurring Yen 612,000 of additional cost. In this way, slight delay in any process often lead to significant impact to the project in the case of software development.

This countermeasure plan requires Worker A to perform one holiday work on July 17 (Saturday), and eliminating the impact of the delay on the subsequent processes. As a result, the cost increase due to delay is limited to Processes DR and DD2. It should be noted that, under this countermeasure plan, Worker A is not allowed to have his/her compensation day within a week from the holiday work. This is because the countermeasure plan assigned Worker A to Process PE on July 20, and July 21, and he/she is not assigned to this project process during the period of July 22, 23, 24 (Worker A is to engage in other tasks). Therefore, his/her holiday work on July 17 (Saturday) is treated as overtime work, which requires 25% or more overtime premium. So we calculated the daily cost as 125% in this case.

Compare the proposed original development plan and countermeasure plan you can find that the project end date is August 2, and that the countermeasure plan cost is Yen 1,340,800. This means that the cost increase was Yen 84,800 or 6.7% from proposed original development plan (as 100%). In other words, this fact demonstrates that the delay was recovered with very little cost.

7 Comparison with Related Work
There are many different models that attempt to present typical work structures of software development, including PMDB [9], Design-Net [10, 11], kyotoDB [12], and PROMX [13]. However, these models do not explicitly address either the relationship between software development tasks and the resources essential to conduct the tasks or the constraints on the conditions and available periods of resource assignment although they are useful for the models to represent the task model of...
a project because they focus on how to represent the hierarchy and sequence of tasks (Although PMDB uses Person as an entity, it does not address the constraints related to the resource allocation conditions and the available period of resources). Therefore, they are not adequate for the project management models of software development projects.

(1) Our viewpoint of man-hours estimation is different from that of CCPM. CCPM adopts man-hours estimation of each process with 50% of success probability and uses the joining and project buffers to reduce the risk of process delay due to estimation errors. In our approach, an average of extra man-hours is calculated for the joining and project buffers and assigned to each process.

(2) Our viewpoint of progress management is different from that of CCPM. CCPM manages the progress of the whole project by examining the consumption ratio of the buffer instead of managing the progress of each process. For this reason, CCPM can be used to detect process delays of the whole project, but it is not adequate for understanding the progress of processes which are not on the Critical Chain. In our approach, progress is managed by each process. As a result, it is possible to understand the progress of every process, regardless of if it is on the Critical Path.

(3) When a process delay is detected, it is not easy to change the project schedule in CCPM to recover the delay, but in our approach as described in this paper and Reference [3, 4, 5], it is possible to use our tool to develop a revised plan dynamically that can be used to recover the process delay.

Finally, we compare the system with the conventional systems [1-7]. The differences between conventional systems are as follows.

(4) In the case of Reference [1], only one worker can be assigned to one process. However, Reference [6] enables multiple worker assignment to a process.

(5) Reference [5] allows the worker to start the subsequent task even before the completion of the preceding task, so that the "constraints on operational sequence" in the cases of Reference [1, 6, 7] are mitigated, and, instead, "constraints on resource assignment conditions" are enhanced. Specifically, when there is a delay in Process A under Worker α, subsequent Process B should be assigned to Worker β with additional requirement of "skill level of Worker β must be equal or higher than that of Worker α."

8 Conclusion
Typical countermeasures against a process delay consist of the following four types and their combinations.

(1) Recover the progress by holiday works
(2) Crashing (additional assignment of excess personnel) [3]
(3) Fast tracking (start the subsequent work before its preceding work has finished.)[5]
(4) Reduction (Omit part of the system from delivery)

In regard to the above mentioned (2) and (3), we have already proposed the mechanisms to implement the respective system and proved the effectiveness. We determined that we will not consider the reduction as our system does not support it. After all, we had (1) left untouched. For this reason, this paper proposes the mechanism to create the countermeasure plan against process delay using holiday works. We focused on the constraints inherent to software development planning problems, and clarified the difference between original development planning, cases (2) and (3) as above mentioned on the basis of different ways of handling the constraints. Finally, we introduced an example in which a process delay can be recovered with a little cost by implementing (1), and proved the effectiveness of the proposed mechanism.

Acknowledgement
We appreciate those who are concerned with Graduate School of Engineering, Shibaura Institute of Technology for their cooperation and support.

References:


