Dynamic Resource Scheduling in Disruption-Prone Software Development Environments

Junchao Xiao[1,2], Leon J. Osterweil[2], Qing Wang[1], and Mingshu Li[1,3]

[1] Laboratory for Internet Software Technologies, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
[2] Department of Computer Science University of Massachusetts, Amherst, MA 01003-9264 USA
[3] Key Laboratory for Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China

Abstract. Good resource scheduling plays a pivotal role in successful software development projects. However, effective resource scheduling is complicated by such disruptions as requirements changes, urgent bug fixing, incorrect or unexpected process execution, and staff turnover. Such disruptions demand immediate attention, but can also impact the stability of other ongoing projects. Dynamic resource rescheduling can help suggest strategies for addressing such potentially disruptive events by suggesting how to balance the need for rapid response and the need for organizational stability. This paper proposes a multi-objective rescheduling method to address the need for software project resource management that is able to suggest strategies for addressing such disruptions. A genetic algorithm is used to support rescheduling computations. Examples used to evaluate this approach suggest that it can support more effective resource management in disruption-prone software development environments.

Keywords: Disruption, rescheduling, multi-objective, genetic algorithm.

1 Introduction

Software development processes are highly dependent upon human resources [1, 25]. Thus a key problem in software project management is appropriate human resource scheduling. Effective resource scheduling should ensure that assigned resources have the capability and capacity to execute their assigned tasks, that resource contention is minimized, project efficiency is maximized, and that organizational value and customer satisfaction are increased [3, 28, 29].

But disruptive events such as requirements changes, needs for fixing important bugs, incorrect or unexpected process execution, and staff turnover can create uncertainty that complicates resource scheduling [13, 21]. A particularly vexing aspect of this problem is how to balance the need to respond effectively to disruptive events against the need to be sure that this response does not create other (perhaps even more severe) disruptions by destabilizing other ongoing projects. There are risk management approaches that suggest how to anticipate and address some kinds of such disruptive events, but unexpected events simply cannot be predicted with sufficient accuracy, thus suggesting the need for a dynamic rescheduling approach.

D.S. Rosenblum and G. Taentzer (Eds.): FASE 2010, LNCS 6013, pp. 107–122, 2010.
© Springer-Verlag Berlin Heidelberg 2010

Answers to the following four questions are needed to support this approach:

(1) Under what circumstances should a rescheduling be executed? That is, how can the problems caused by disruptions and the current state of the process execution be used to suggest when a rescheduling should be undertaken?
(2) Which activities should be covered by the rescheduling? That is, what activities should be within the scope of the rescheduling when rescheduling is undertaken?
(3) How can the approach to rescheduling be tailored to accommodate different kinds of disruptions? What kinds of measures can be used to support finding a balance between dealing with current disruptions and avoiding the creation of excessive new disruptions in doing so?
(4) What kind of scheduling algorithm should be used? Which algorithm can provide as optimal a scheduling result as possible for costs in time and computing power that are as minimal as feasible?

This paper proposes a software process rescheduling method to address the issues of software project management in these kinds of dynamic disruption-prone environments. Articulate process and resource models are used to support this method. The value obtained from proposed reschedulings is computed using a function that weights both how well the rescheduling addresses the disruption (utility) and how little it creates new disruptions (stability). To address the problems posed by the high degree of complexity of such a rescheduling problem, a genetic algorithm (GA) [14] is adopted as the basis for our rescheduling approach.

The paper provides the following contributions:

(1) **A procedure for performing resource rescheduling in response to the occurrence of disruptive events** that assumes pre-specified responses to disruptions, and tackles the problems caused by executing these responses.
(2) **A multi-objective value function for evaluating rescheduling results** that takes into consideration the need for both high stability and high utility.
(3) **A GA based rescheduling method that seems to be effective** both in delivering good results and efficient in keeping costs modest.

Section 2 analyzes uncertainties and process change in software development, and describes the rescheduling approach we use to address disruptions. Section 3 presents the models used as the bases for both the rescheduling and the evaluation of the multi-objective function used to evaluate the rescheduling. Section 4 provides an initial demonstration of this method. Section 5 describes some related work, and section 6 presents conclusions and suggests future work.

2 A Rescheduling Approach to Responding to Dynamic Change

Software development managers need to make resource rescheduling decisions to respond to disruptions [15] such as:

(1) Requirement velocity: Requirements continually change during process executions [12, 16]. To address these changes, new activities may be inserted into development processes, requiring assigning resources to these activities.

(2) Sudden arrival of urgent activities: New activities may be needed to address urgent problems (e.g. serious bugs in delivered software) [20]. Although such events might not be unexpected, it may be hard to predict when they occur, and thus the changes they require may cause disruption to schedule or cost [5].
(3) Deviations in process execution: Inaccuracies in project cost estimates, incorrect performance of tasks by project personnel, the unexpected need for rework, or the occurrence of process exceptions may cause a project to fail to proceed as planned thus necessitating the rescheduling of project resources.
(4) Staff turnover: The software industry experiences high personnel mobility and staff turnover that create disruptions that typically require rescheduling [7].

This paper presents a resource rescheduling method designed to tackle the impact of these kinds of disruptions in dynamic environments. The first step in our method entails determining the changes to process execution needed to respond to the occurrence of disruptive events. These changes may include either the insertion of new process activities, the deletion of activities that were present in the initial process, or the addition or deletion of resources that had been in the initial resource set. Any of these changes triggers resource rescheduling.

The second step entails determining the scope of the resource rescheduling, namely identification of the activities and resources to be involved in a rescheduling. There are a number of reasons why rescheduling does not necessarily entail reconsideration of all the activities and resources in an entire software organization. One such reason is that process changes may occur in only one project or even in only a part of one project, and the needed resource scheduling adjustments might be readily restricted to this range. Moreover, we note that if rescheduling spans the entirety of a long term process, the rescheduling might itself introduce more disruptions than it addresses. In this paper the scope of a rescheduling is restricted to only a subset of the projects being performed by an organization at the time of the rescheduling.

Third, we construct the constraints and value objectives for rescheduling. The goal of rescheduling is to obtain optimal organizational value while conforming to various constraints. Since resource scheduling decisions are usually made under conflicting goals, a value function that can balance the goals is needed in rescheduling. This paper uses as an example addressing the conflicting goals of stability and utility.

Fourth, we seek value function optimization by using GA. This optimization problem has a high level of complexity, so we use GA as the scheduling approach, hoping to achieve near-optimal results at acceptable costs.

Section 3 describes our approaches to the problems arising in doing these steps.

3 Multi-objective Resource Rescheduling Using a GA

3.1 Project, Activity and Resource Models Used in Scheduling

3.1.1 Project Model

Software organizations are usually performing a group of projects, each described by basic information, constraints, and its value objectives. Thus we define a project as:

Definition 1. $P = (BasicAttr, ConSet, PWSet)$ **, where,**

- *BasicAttr* describes such basic attributes as name, generation, descriptions, etc.
- *ConSet* is the set of all project level constraints (e.g. cost and time). Violation of each constraint will incur some quantified penalty.
- *PWSet* is a preference weight set. Each project in a multi-project environment is assigned a priority weight relative to the other projects. This weight is used to evaluate the importance of the resource requirements of each project. Note that this weight may change dynamically (e.g. to emphasize the importance of responding to the need to fix an important bug).

3.1.2 Activity Model

Precision and specificity in evaluating competing resource schedules are enhanced through the use of a project specification notation that is more precise and detailed. Thus, the Little-JIL process definition language [26] is used in this paper to define software development project activities, their dependencies upon each other, and their needs for resources. This language offers simplicity, semantic richness and expressiveness, and a formal and precise, yet graphical, syntax.

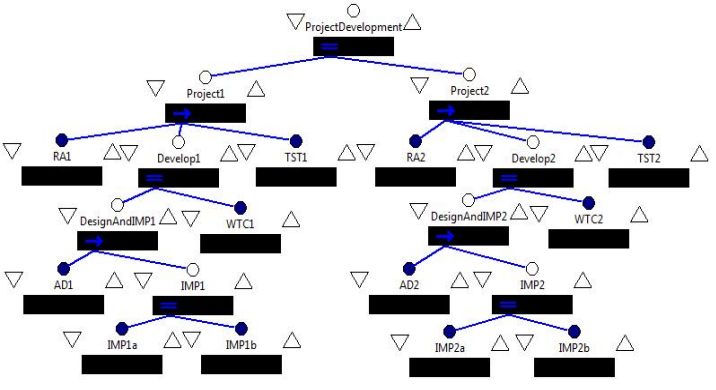

Fig. 1. Project development process described by Little-JIL

Fig. 1 shows a Little-JIL definition of a process for carrying out two software development projects in parallel. The "=" sign in the "ProjectDevelopment" root step indicates that its substeps (Project1 and Project2) are performed in parallel. Projects are decomposed into requirement analysis (RA), development (Develop), and testing (TST), which are executed sequentially (represented by "→" sign). Development is further decomposed into design (AD), implementation (IMP), and write test case (WTC). WTC is executed parallel with AD and IMP, which are executed sequentially.

Requests for resources are represented iconically by the dot atop the step, and are described as required skills, skill levels, and required quantities of skills.

Fig. 2 shows an example of how an activity may be added to a process to respond to a disruption. Thus Project3 executes RA, IMP, and TST sequentially to realize a changed requirement and Project4 and Project5 use IMP and TST sequentially to

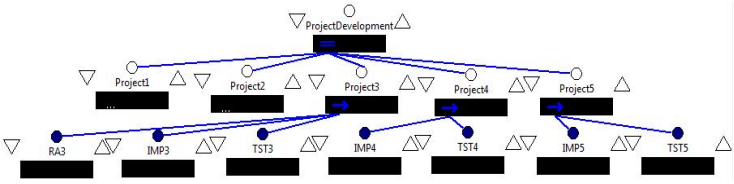

Fig. 2. Combine new project activities with initial project development process

realize urgent bug fixing. All are executed in parallel with Project1 and Project2. This process definition is used to select the activities that will be included in the scope of the rescheduling. This is done using graph searching algorithms whose details are omitted due to space constraints. In the work described here, all steps that are fewer than some previously fixed number of steps beyond the currently executing step(s) are included in the rescheduling. Details of how this is done can be found in [27]. Other criteria could be used to determine the steps in this rescheduling "window".

3.1.3 Resource Model

The human resource model proposed in [29] is used to describe human resources. Each human resource is described by its identification (ID), executable activity type set (EATS), skill set (SKLS), experience data (EXPD), schedulable time and work-load (STMW), together with salary per person-hour (SALR). EATS, SKLS and EXPD are capability attributes, and STMW is the capacity attribute.

Resources available for the rescheduling are preserved in a resource repository. The set of resources that are candidates to perform a step are those such that 1) the work type of the step is included in the EATS attribute of the resource, 2) the skills described in the step's requirements are also in the resource's SKLS attribute, and 3) the resource has a higher skill level than is required by the step. Note that when there is resource turnover, the resource repository is changed accordingly.

3.2 Multi-objective Value Measure of Rescheduling

A rescheduling may have different objectives that conflict with each other. Thus, for example, attaching a high priority to fixing a bug in one project indicates that this bug fix will return high value to the organization. But this may require using the resources of a different project, causing disruption to that project and loss of value to the organization. Thus rescheduling must be measured against a possible multiplicity of objectives. Assume a rescheduling has n objectives, $o_1, o_2, ..., o_n$, each having weight w_i, then the value of the rescheduling is defined to be $MO = \sum_{i=1}^{n} w_i * o_i$

Note that each objective in this function can be further decomposed into sub-objectives, each having its own importance weight. This paper presents, as an example, two resource rescheduling objectives to define the value function. The first (stability) weights the importance of keeping the rescheduled process similar to the initial process. The second (utility) weights the importance of responding to the

disruption. Though the examples in this paper assume the existence of only two objectives, the approach scales up to consideration of any number of objectives.

3.2.1 Stability Value

Process stability is measured using two factors, change in the scheduling of each activity and change in human resource assignments to each of the activities. Schedule changes can reduce project commitment and customer satisfaction. Resource assignment changes can necessitate more communication effort, more training time, and waste of previous preparations, reducing the value of a project.

Schedule deviation is measured by the differences between initial process and rescheduled process start times and end times. Let the start time before and after rescheduling of an activity *ACT* be ts_{ACT} and ts'_{ACT} respectively, the end time of *ACT* be te_{ACT} and te'_{ACT} respectively. Since impact of start time and end time deviations may differ, let impact coefficients be α and β respectively. Then the deviation of *ACT* is defined to be $\alpha * \left| ts_{ACT} - ts'_{ACT} \right| + \beta * \left| te_{ACT} - te'_{ACT} \right|$, and the total deviation of the activities in *ActivitySet* due to rescheduling is:

$$SDeviation = \sum_{\text{ACT} \in ActivitySet} (\alpha * \left| ts_{ACT} - ts'_{ACT} \right| + \beta * \left| te_{ACT} - te'_{ACT} \right|)$$

Note that in this example only the activities in the initial process are used to compute schedule deviation. Section 3.1.2 suggest how these activities are identified, and more complete details can be found in [27]. Other measures can also be defined.

To measure human resource changes, workload changes for each human resource scheduled to an activity are accumulated. Assume a human resource set HRS represents all the human resources assigned to an activity *ACT* in either the initial or the rescheduled process. For each hr ∈ *HRS*, assume the workload allocated to *ACT* before rescheduling is E^b_{hr}, and after rescheduling is E^a_{hr}. If hr is not in *ACT* before rescheduling, E^b_{hr} is zero. If hr is not in *ACT* after rescheduling, E^a_{hr} is zero. Then total human resource changes are defined to be: $HRDeviation = \sum_{\text{hr} \in HRS_{ACT}} \left| E^b_{hr} - E^a_{hr} \right|$

Schedule deviation and resource change may have different impacts on stability, and so *HC*, a coefficient of human resource change is used to compute total deviation.

$$Deviation = \sum_{\text{ACT} \in ActivitySet} (\alpha * \left| ts_{ACT} - ts'_{ACT} \right| + \beta * \left| te_{ACT} - te'_{ACT} \right| + HC * \sum_{\text{hr} \in HRS_{ACT}} \left| E^b_{hr} - E^a_{hr} \right|)$$

The goal of stable rescheduling is to minimize the above stability loss. Therefore, total stability value is *StabilityValue = C-DP*Deviation*, where DP is the deviation penalty coefficient and constant C causes the stability value to be positive.

3.2.2 Utility Value

Utility describes the value obtained from a project that satisfies its constraints at its conclusion. If the project succeeds and satisfies its constraints, benefits will be obtained. If the project is delayed, penalties are incurred. The schedule utility of a project is defined by comparing the actual finishing date to the constraint finishing date. Let the actual finishing date and the constraint finishing date of a project be *AFD* and

CFD respectively. Let benefit of finishing one day ahead the constraint be *SB,* and the penalty for a one day delay be *SP*. The schedule utility is defined to be:

$$SUtility = SB * \max\{(AFD - CFD),0\} - SP * \max\{(CFD - AFD),0\}$$

Since the cost of developers is the primary cost in software development, our method only takes human resource cost into consideration. This cost is the total over all activities of the product of the salary rate of each human resource multiplied by the workload required. Assume the cost of a project is *CST* and cost constraint of this project is *CCST*, then cost utility of this project is: $CUtility = CCST - CST$

Weighting schedule and cost preference by coefficients *SWeight* and *CWeight* respectively, project utility is: $PU = SWeight * SUtility + CWeight * CUtility$

The preference weights of projects vary. For example, an urgent bug fix project may be very important and should have a high priority for resources. Thus a project preference weight (PPW) is set for each project and the utility value for all projects in an organization is defined by: $UtilityValue = \sum_{P \in ProjectSet} (PPW_P * PU_P)$

Now finally, assume the stability and utility objectives for a scheduling are given weights w_s and w_u respectively. Then a rescheduling's value is computed by:

$$Value = w_s * StabilityValue + w_u * UtilityValue$$

3.3 Rescheduling Using a GA

3.3.1 Encoding and Decoding

The first step in using GA as a problem solver is to represent the problem as a chromosome. In the activity model described by Little-JIL, non-leaf steps are used to represent scopes and to group certain kinds of activities, but only leaf steps represent actual project performance activities. Thus, once the scope of rescheduling has been determined as described in section 3.1.2 (and in more detail in [27]), only the leaf steps in that scope are selected for GA encoding. Assume the *N* steps, $S_1, S_2, ..., S_N$ are selected, and the human resources capable of executing step S_i are $HR_{i,1}$, $HR_{i,2}$, ..., $HR_{i,ti}$. We construct a resource queue $HR_{1,1}$, $HR_{1,2}$, ..., $HR_{1,t1}$, ..., $HR_{N,1}$, ..., $HR_{N,,tN}$, consisting of all resources that are schedulable to activities in the rescheduling scope. The first part of the chromosome (shown as the left part of Fig. 3) is generated by creating a gene for each step, as just described. The length of this part is: $T = \sum_{i=1}^{N} t_i$.

Once GA has run, if a gene has value "1" the corresponding human resource has been scheduled to the corresponding step. The value "0" means the corresponding human resource is not scheduled to the step.

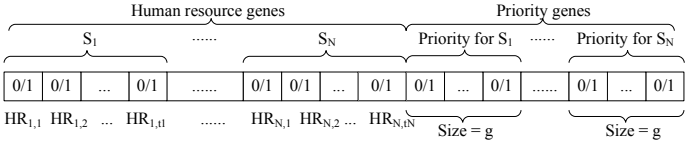

Fig. 3. Structure of the Chromosome

The chromosome also contains priority genes (shown on the right of Fig. 3) to represent the priority weight of each project. A priority weight is a binary number. If the GA assigns a human resource to more than one step, the step with highest priority value is assigned the human resource. Therefore, the length of the chromosome is:

$CL = T + N * g$, where g is the base 2 logarithm of the maximum priority level.

After GA has been run, a chromosome is decoded into a schedule as follows. First, sort all the steps involved in the rescheduling into a queue. In this queue, steps that precede others are placed in front of those that follow. If steps do not have a precedent/succedent relationship, steps with higher priority are placed in front of those with lower priority. Second, assign each resource whose gene has the value "1" to the corresponding step in the queue. If a step requires only a certain number of resources, then at most that number of resources are assigned. Third, allocate the schedulable workload of the assigned resources to each step and update the availability state of the resources. Finally, set the start time of the step so that it is the earliest time that is not earlier than the end time of all of its preceding steps.

***Constraint satisfaction*:** Rescheduling constraints are built into the encoding and decoding process. During encoding, candidate resources for each step are determined to have the capability to execute the step. In the decoding process, only resources that have available workload are scheduled, and they are scheduled only at times when they are available and when the activity is actually executable. Other constraints could also be defined and used to assure that more qualified resources (e.g. those with higher skill levels) are favored for assignment over less qualified resources.

3.3.2 Running GA

The initial population of the GA is generated by creating chromosomes as described above. Evolution is realized by using predefined crossover and mutation rates, for each population generation. The fitness of each chromosome is evaluated by the value function presented in section 3.2 and chromosomes with higher fitnesses are selected for each succeeding generation. Evolution continues for a predetermined number of generations, and the chromosome with the highest fitness in the last generation is selected as the solution. Full details of the steps used in executing GA are omitted here due to space limitations. These details can be found, however, in [28].

4 Demonstration of the Use of This Approach

To evaluate our method, we used it to simulate the allocation of resources by a software company engaged in two different projects. We hypothesize that each of the two projects is addressing requirements for a group of modules, and that both are doing so by performing the process shown in Fig. 1.

Resources available to the company are listed in Table 1 and the leaf activities of the two projects are described in Table 2. Due to space constraints, in the human resource description in Table 1 we show only productivity (obtained from experience data) and salary rates for each resource. We assume human resources are available only on workdays from 1 January 2009 through 31 December 2010 and each workday

Table 1. Human resource information of initial process

Human resource	Executable activity and corresponding productivity (KLOC/Person-Hour)	Salary rate (RMB)	Human resource	Executable activity and corresponding productivity (KLOC/Person-Hour)	Salary rate (RMB)
HR1	RA/0.06	60	**HR8**	IMP/0.025	40
HR2	RA/0.04	45	**HR9**	IMP/0.02	35
HR3	RA/0.05	50	**HR10**	IMP/0.015	35
HR4	AD/0.06	60	**HR13**	WTC/0.05; TST/0.04	45
HR5	AD/0.05	60	**HR14**	WTC/0.045; TST/0.035	45
HR6	AD/0.05	50	**HR15**	WTC/0.035; TST/0.03	45
HR7	IMP/0.03	45	**HR16**	WTC/0.03; TST/0.03	40

Table 2. Activity information of initial process

Activity	Candidate resources	Size (KLOC)	Initial allocated resources	[Start, End]
RA1	HR1, HR2, HR3	20	HR1, HR2	[2009-05-01, 2009-06-05]
AD1	HR4, HR5, HR6	20	HR4, HR5	[2009-06-08, 2009-07-08]
IMP1a	HR7, HR8, HR9, HR10	12	HR7, HR8	[2009-07-09, 2009-08-17]
IMP1b	HR7, HR8, HR9, HR10	8	HR9, HR10	[2009-07-09, 2009-08-18]
WTC1	HR13, HR14, HR15, HR16	20	HR13, HR14	[2009-06-08, 2009-07-23]
TST1	HR13, HR14, HR15, HR16	20	HR13, HR14	[2009-08-19, 2009-10-5]
RA2	HR1, HR2, HR3	16	HR1, HR3	[2009-05-21, 2009-06-24]
AD2	HR4, HR5, HR6	16	HR5, HR6	[2009-06-25, 2009-07-30]
IMP2a	HR7, HR8, HR9, HR10	10	HR7, HR8	[2009-08-18, 2009-09-17]
IMP2b	HR7, HR8, HR9, HR10	6	HR9, HR10	[2009-08-19, 2009-09-17]
WTC2	HR13, HR14, HR15, HR16	16	HR15, HR16	[2009-06-25, 2009-08-11]
TST2	HR13, HR14, HR15, HR16	16	HR15, HR16	[2009-09-18, 2009-11-4]

has 8 person-hour workloads available. In the activity description shown in Table 2, we show only the candidate resources, size, initial allocated resources, and the start and end time for each activity execution. The candidate resources are identified by matching activity resource requests to human resource capabilities.

We now assume that after resources have been scheduled to the projects' activities three new requirements are issued. One is an upgrade requirement that is addressed by the process specified as Project3 in Fig. 2, and the other two are to address the sudden arrival of urgent bug fixing requests to be done as specified by Project4 and Project5 in Fig. 2. Leaf activities of these projects are described in Table 3.

Table 3. Activity information of added process

Activity	Candidate resources	Size (KLOC)
RA3	HR1, HR2, HR3	14
IMP3	HR4, HR5, HR6	14
TST3	HR13, HR14, HR15, HR16	14
IMP4	HR7, HR8, HR9, HR10	10
TST4	HR13, HR14, HR15, HR16	10
IMP5	HR7, HR8, HR9, HR10	8
TST5	HR13, HR14, HR15, HR16	8

Rescheduling is required in order to provide resources to address these new requirements. For this example we assume that rescheduling parameters of the multi-objective value function are set as shown in Table 4 and parameters used for computing project utility are set as shown in Table 5.

Table 4. Parameters of multi-objective value function

α/β/ HC	1 / 1 / 1
C	600,000
DP	300

Table 5. Parameters of projects used for computing utility

	Project1	**Project2**	**Project3**	**Project4**	**Project5**
Constraint start and finish date	[2009-05-01, 2009-10-30]	[2009-05-21, 2009-11-31]	[2009-06-01, 2009-8-31]	[2009-07-01, 2009-9-20]	[2009-08-01, 2009-10-30]
Constraint cost	200000	150000	120000	90000	90000
Schedule benefit/penalty	100 / 200	100 / 200	200 / 400	200 / 400	200 / 400
Schedule/Cost weight	1 / 1	1 / 1	2 / 1	2 / 1	2 / 1
Project weight	1	1	2	3	4

For the GA examples presented here, we set population scale to 60, crossover rate to 0.8, mutation rate to 0.02, and generation number to 500.

4.1 The Need for Rescheduling

We begin by computing an initial resource assignment plan for Project1 and Project2 assuming that there will be no disrupting events. This plan is shown in Fig. 4.

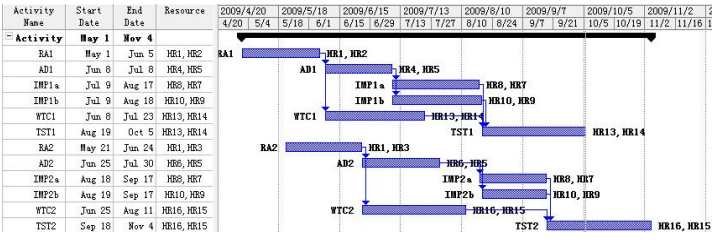

Fig. 4. Initial scheduling result

We then hypothesize the need to provide resources for Project3 starting on 1 June 2009. Activity RA3 has three candidate resources, and let us assume that it is decided that HR1 and HR2 are to be allocated to it. Thus its execution would require 18 workdays, from June 1 to June 24. Then IMP3 would need to be executed starting on June 25. IMP3 has four candidate resources and let us further assume that HR7 and HR8 are selected. Thus IMP3's execution would require 32 workdays, from June 25 to August 7. Finally, TST3 would need to be executed starting on August 8. However, note that from June 1 to June 5, HR1 and HR2 are occupied by RA1 and from June 1 to June 24, HR1 and HR3 are occupied by RA2, thus RA3 could not obtain the resources it needs without disrupting other projects. In addition, from July 9 to September 17, all resources able to execute IMP3 are occupied performing Project1 and Project2. Thus either Project3 must wait or other projects must be disrupted. Organizational value is lost in either case. Therefore, a rescheduling is indicated.

Once the scope of rescheduling has been determined, all the activities in this scope will be encoded in the chromosome, and used in the GA based rescheduling.

4.2 Results under a Specific Stability and Utility Weight Configuration

We start exploring the efficacy of our approach by examining the consequences of two rescheduling approaches, where stability is the only objective, and where utility is the only objective. Fig. 5 shows the rescheduling plan where stability is the only consideration (stability and utility weights are set to 1 and 0 respectively). The start/end times of activities in Project1 and Project2 are not changed, nor are scheduled human resources. The new added activities are executed only when resources are available, causing delay and low utility for Project3.

Fig. 6 shows the rescheduling plan where only utility value is considered (the stability and utility weights are set to 0 and 1 respectively). This schedule causes Project3 to have higher utility, but the start/end times and scheduled resources of most activities in Projects 1 and 2 are changed causing substantial reduction in organizational value. This case study and others not shown due to space constraints indicate that our approach supports scheduling resources to address stability and utility objectives. We now suggest how this capability can help project managers.

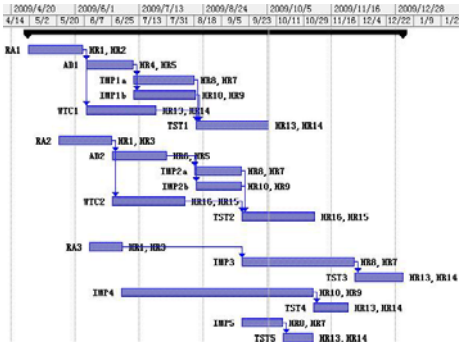

Fig. 5. Rescheduled plan when stability is 1

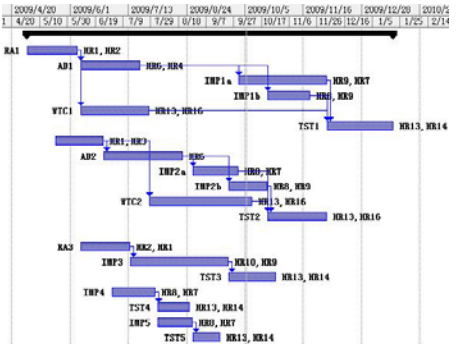

Fig. 6. Rescheduled plan when stability is 0

4.3 Results under Different Stability and Utility Weights

Using a multi-objective value function to evaluate rescheduling can help support exploring the way that different balances between stability and utility can affect organizational value. To demonstrate this we varied the stability and utility weights for a series of reschedulings. Fig. 7 shows the different values of the schedules obtained. As expected increasing the stability weight causes a consequent increase in stability value while utility value decreases. Conversely increasing the utility weight causes utility value to increase while decreasing stability value. Of perhaps more interest, however, is that the maximum total of the two values is obtained over a broad range of stability weights, and dips only when the stability weight is near either 0 or 1. This suggests that moderation in addressing disruption is likely to be the best course of

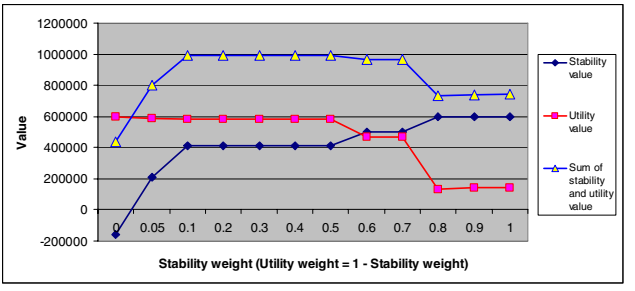

Fig. 7. Stability and utility value under different stability and utility weights (results are averages of 10 different simulations)

action, but the precise relatively weighting of the importance of stability and utility may not be particularly important.

4.4 A Series of Reschedulings to Tackle Multiple Disruptions

The foregoing suggests that our approach could help in deciding how to deal with a single disruptive event. But most software organizations experience a continuing flow of disruptions. Thus next we used our approach to seek a strategy for dealing with such sequences of disruptions. To do this we ran three sets of simulations using the same stability and utility weight combination, but different strategies for handling the disruptions. The first strategy involves one rescheduling, done on June 1, with all three projects initiated simultaneously. The second strategy involves two reschedulings, one on June 1 when Project3 is initiated; and the second on July 1 when Project4 and Project5 are initiated. The third strategy involves initiating one new project on June 1, one on July 1, and one on August 1. We computed utility values for each strategy, where for each we increased the stability weight from 0 to 1 in increments of one tenth. Fig. 8 suggests that none of the approaches seems to offer clear advantages over the others, but that for all total value remains high, and roughly constant when the stability value increases from 0.0 to 0.5, but drops sharply at values higher than 0.5. This result

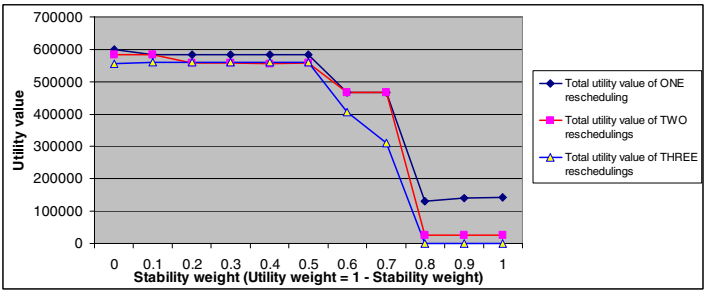

Fig. 8. Utility value curves for three different rescheduling strategies (results are averages of 10 different simulations)

seems to confirm the importance of responding promptly to disruptions, but also suggests that a "drop everything" approach seems to offer little advantage over a more measured response.

More generally, this case study seems to us to indicate that our rescheduling approach offers promise of being a useful tool in formulating and analyzing various strategies for dealing with various kinds and scenarios of disruption.

4.5 Discussion

The above case studies show that our proposed method can provide useful rescheduling results using different combinations of objectives, some of which might entail conflicts. Project managers can compare these results and use them to support their resource allocation decisions by suggesting alternatives to decisions the might have been planned by the human manager.

Currently our method still has some limitations. Some of them are:

The scale of the scheduling problem affects the running time and may make it impractical or impossible to obtain optimal results. If the problem is too large, dividing it into several smaller ones might help.

Different GA parameter settings affect whether optimal or near-optimal rescheduling results can be obtained. Different scheduling problems may suggest the use of different GA parameters.

Additional types of constraints could be used to improve the quality of schedules.

5 Related Work

Researchers have observed that software development project disruptions can be due to uncertainties in requirements, process execution, and human resources. Ebert et al. [1] analyze requirement uncertainties and provide problems that are caused by them. Li et al. [16] use rough set analysis to solve problems caused by requirement uncertainties. Pfahl et al. [22] and Liu et al. [17] use simulation to explain some effects arising from requirements volatility. Cass et al. [8] indicate that rework is an ongoing problem in software development. Melo et al. [19] point out that resource change is a key factor in maintenance schedule postponement and cost overrun. Dynamic rescheduling is not suggested as a way to deal with any of these different kinds of uncertainties.

Software process scheduling has been explored by quite a few researchers. Some methods provide schedules that are based upon the assumption of accurate human resource specifications, such as skills, productivity, and availability, and are thus able to satisfy constraints and obtain optimal scheduling values [3, 6, 9, 11]. However, these methods only address scheduling problems that arise in response to specifically anticipated activities and resource changes. Unlike our approach, they cannot dynamically respond during actual process execution to disruptions caused by unanticipated events.

To address uncertainties in software development and maintenance processes, Antoniol et al. [4] present a scheduling method that combines a genetic algorithm and queue simulation. Though the method realizes scheduling under some uncertainties,

issues such as stability are not taken into account. Other methods tackle uncertainties by introducing probability into scheduling. Liu et al. [18] suggest a probability based two stage scheduling method. Though the method uses probability of commitment satisfaction in scheduling, dynamic changes still lead schedule disruption.

There are a lot of rescheduling methods in the manufacturing domain [2, 10, 23, 24]. These methods use rescheduling to achieve both makespan and stability value. However, the resources in manufacturing are usually machines, which do not pose problems such as volatility and skill level change that are characteristics of human resources. This limits the applicability of this work to software development

6 Conclusions and Future Work

This paper has presented a multi-objective software process resource rescheduling method using a GA. We identified some conditions that can necessitate rescheduling and introduced models to describe projects, activities, and resources. We then used these models to define some rescheduling problems, and presented a multi-objective value function that weights stability and utility to compute rescheduling value. The evaluation of our method shows that this multi-objective value function can be used to guide rescheduling, and might help managers to balance potentially conflicting objectives in making resource rescheduling decisions.

Future work

Continuous rescheduling: In our case study, we used examples for which relatively accurate parameter estimates are available. However, such estimates usually change during process execution. Thus process delay and activity completion date changes happen frequently. In future work, such continuous changes and reschedulings will be taken into consideration.

More constraints: This paper only models capability constraints, availability constraints, and activity execution order constraints in scheduling. Other constraints, such as different activities needing the same resource, will be modeled in future work.

Analysis of different objectives and their importance: The activities on a critical path have more impact on the stability of a project. Thus it seems more critical to schedule these activities than to schedule other activities. Furthermore, there are many other kinds of objectives in rescheduling. In future work, more objectives and more details of these objectives will be analyzed and used in rescheduling.

Acknowledgments. This paper was supported by the National Natural Science Foundation of China under grant No. 90718042, the 863 Program of China under grant No. 2007AA010303, 2007AA01Z186, as well as the 973 program under grant No. 2007CB310802. This work was also supported by the National Science Foundation under Awards No. CCR-0205575, CCR-0427071, and IIS-0705772. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

1. Acuña, S.T., Juristo, N., Moreno, A.M.: Emphasizing Human Capabilities in Software Development. IEEE Software 23(2), 94–101 (2006)
2. Adibi, M.A., Zandieh, M., Amiri, M.: Multi-objective scheduling of dynamic job shop using variable neighborhood search. Expert Systems with Applications 37(1), 282–287 (2009)
3. Alba, E., Chicano, J.F.: Software Project Management with GAs. Journal of Information Sciences 177(11), 2380–2401 (2007)
4. Antoniol, G., Penta, M.D., Harman, M.: A Robust Search–Based Approach to Project Management in the Presence of Abandonment, Rework, Error and Uncertainty. In: Proceedings of the 10th International Symposium on Software Metrics, pp. 172–183 (2004)
5. Antoniol, G., Penta, M.D., Harman, M.: Search–Based Techniques Applied to Optimization of Project Planning for a Massive Maintenance Project. In: Proceedings of the 21st International Conference on Software Maintenance, pp. 240–249 (2005)
6. Barreto, A., Barros, M.d.O., Werner, C.M.L.: Staffing a software project: A con-straint satisfaction and optimization-based approach. Computer & Operations Research 35(10), 3073–3089 (2008)
7. Boehm, B.: Software risk management: principles and practices. IEEE Software 8(1), 32–41 (1991)
8. Cass, A.G., Sutton Jr., S.M., Osterweil, L.J.: Formalizing Rework in Software Processes. In: Oquendo, F. (ed.) EWSPT 2003. LNCS, vol. 2786, pp. 16–31. Springer, Heidelberg (2003)
9. Chang, C.K., Jiang, H.-y., Di, Y., Zhu, D., Ge, Y.: Time-line based model for software project scheduling with genetic algorithms. Information and Software Technology 50(11), 1142–1154 (2008)
10. Cowling, P., Johansson, M.: Using real time information for effective dynamic scheduling. European Journal of Operational Research 139(2), 230–244 (2002)
11. Duggan, J., Byrne, J., Lyons, G.J.: Task Allocation Optimizer for Software Construction. IEEE Software 21(3), 76–82 (2004)
12. Ebert, C., Man, J.D.: Requirements Uncertainty: Influencing Factors and Concrete Improvements. In: Proceedings of the 27th International Conference on Software Engineering, pp. 553–560 (2005)
13. Erdogmus, H., Favaro, J., Halling, M.: Valuation of Software Initiatives Under Uncertainty: Concepts, Issues, and Techniques. In: Biffl, S., Aurum, A., Boehm, B., Erdogmus, H., Grünbacher, P. (eds.) Value-Based Software Engineering, pp. 39–66 (2005)
14. Holland, J.H.: Adaptation in natural and artificial systems. MIT Press, Cambridge (1992)
15. Li, M., Yang, Q., Zhai, J., Yang, G.: On Mobility of Software Processes. In: Wang, Q., Pfahl, D., Raffo, D.M., Wernick, P. (eds.) SPW 2006 and ProSim 2006. LNCS, vol. 3966, pp. 105–114. Springer, Heidelberg (2006)
16. Li, Z., Ruhe, G.: Uncertainty Handling in Tabular-Based Requirements Using Rough Sets. In: Ślęzak, D., Yao, J., Peters, J.F., Ziarko, W.P., Hu, X. (eds.) RSFDGrC 2005. LNCS (LNAI), vol. 3642, pp. 678–687. Springer, Heidelberg (2005)
17. Liu, D., Wang, Q., Xiao, J., Li, J., Li, H.: RVSim: A Simulation Approach to Predict the Impact of Requirements Volatility on Software Project Plans. In: Wang, Q., Pfahl, D., Raffo, D.M. (eds.) ICSP 2008. LNCS, vol. 5007, pp. 307–319. Springer, Heidelberg (2008)

18. Liu, X., Yang, Y., Chen, J., Wang, Q., Li, M.: Achieving On-Time Delivery: A Two-Stage Probabilistic Scheduling Strategy for Software Projects. In: Wang, Q., Garousi, V., Madachy, R., Pfahl, D. (eds.) ICSP 2009. LNCS, vol. 5543, pp. 317–329. Springer, Heidelberg (2009)
19. de Melo, A.C.V., de J. Sanchez, A.: Bayesian Networks in Software Maintenance Management. In: Vojtáš, P., Bieliková, M., Charron-Bost, B., Sýkora, O. (eds.) SOFSEM 2005. LNCS, vol. 3381, pp. 394–398. Springer, Heidelberg (2005)
20. Moratori, P., Petrovic, S., Vazquez, A.: Match-Up Strategies for Job Shop Rescheduling. In: Nguyen, N.T., Borzemski, L., Grzech, A., Ali, M. (eds.) IEA/AIE 2008. LNCS (LNAI), vol. 5027, pp. 119–128. Springer, Heidelberg (2008)
21. Ozdamar, L., Alanya, E.: Uncertainty Modelling in Software Development Projects (With Case Study). Annals of Operations Research 102, 157–178 (2001)
22. Pfahl, D., Lebsanft, K.: Using Simulation to Analyze the Impact of Software Requirements Volatility on Project Performance. Information and Software Technology 42(14), 1001–1008 (2000)
23. Pfeiffer, A.S., Kadar, B., Monostori, L.S.: Stability-oriented evaluation of rescheduling strategies, by using simulation. Computers in Industry 58(7), 630–643 (2007)
24. Rangsaritratsamee, R., F Jr., W.G., Kurz, M.B.: Dynamic rescheduling that simultaneously considers efficiency and stability. Computers & Industrial Engineering 46(1), 1–15 (2004)
25. Wang, Q., Xiao, J., Li, M., Nisar, M.W., Yuan, R., Zhang, L.: A Process-Agent Construction Method for Software Process Modeling in SoftPM. In: SPW/ProSim 2006, Shang-hai China, pp. 204–213 (2006)
26. Wise, A.: Little-JIL 1.5 Language Report Department of Computer Science, University of Massachusetts, Amherst UM-CS-2006-51 (2006)
27. Xiao, J., Osterweil, L.J., Wang, Q., Li, M.: Dynamic Scheduling in Systems with Complex Resource Allocation Requirements. Department of Computer Science at the Uni-versity of Massachusetts Amherst. Technical report: UM-CS-2009-049 (2009)
28. Xiao, J., Wang, Q., Li, M., Yang, Q., Xie, L., Liu, D.: Value-based Multiple Software Projects Scheduling with Genetic Algorithm. In: Wang, Q., Garousi, V., Madachy, R., Pfahl, D. (eds.) ICSP 2009. LNCS, vol. 5543, pp. 50–62. Springer, Heidelberg (2009)
29. Xiao, J., Wang, Q., Li, M., Yang, Y., Zhang, F., Xie, L.: A Constraint-Driven Human Resource Scheduling Method in Software Development and Maintenance Process. In: Proceedings of 24th International Conference on Software Maintenance, pp. 17–26 (2008)