Modeling Human Decision Behaviors for Accurate Prediction of Project Schedule Duration

Sanja Lazarova-Molnar and Rabeb Mizouni,

Faculty of Information Technology, United Arab Emirates University,
PO Box 17551, Al Ain, United Arab Emirates
{sanja, mizouni}@uaeu.ac.ae

Abstract. Simulation techniques have been widely applied in many disciplines to predict duration and cost of projects. However, as projects grew in size, they also grew in complexity making effective project planning a challenging task. Despite several attempts to achieve accurate predictions, simulation models in use are still considered to be oversimplified. They often fail to cope with uncertainty due to the complex modeling of the high number of interrelated factors. In this paper we propose a simulation model to cope with human resources uncertainty. We use the proxel-based simulation method to analyze and predict duration of project schedules exhibiting high uncertainty and typical human resources reallocation. The proxel-based simulation is an approximate simulation method that is proven to be more precise than discrete-event simulation. To model uncertainty, we introduce a new type of task, state-dependent (floating) task that supports and demonstrates a high degree of uncertainty in human resources allocation. In fact, it allows attributing different probability distributions to the same activity, depending on the team that may perform it. We use software development scheduling to illustrate our approach.

Keywords: Project scheduling, simulation, uncertainty, human resource allocation, on-the-fly decisions.

1 Introduction

Project planning is a discipline that is receiving a constantly growing attention. It is mainly because it has been identified as one of the success/failure factors of a project

As projects grow in size, they also grow in complexity making effective project planning a hard task. It is mainly due to several interrelated factors that all have to be taken into consideration to predict in an accurate and precise way the cost and the duration of a project [1].

Many attempts have been conducted to improve the project scheduling problem [2-6], and each of them endeavors to offer an optimized schedule for a given project. Simulation is one of the techniques that has been successfully applied to project planning, e.g. in construction [7], where they use combined discrete-event/continuous simulation.

Duration of tasks in many project schedules cannot be modeled as deterministic [8], i.e. using fixed numbers. Consequently, probability distribution functions need to be utilized to describe durations of tasks. Even with this assumption, the obtained simulation models are still considered limited and nonrealistic. They fail to take into account the different interrelated factors and uncertainty that in practice lead to plan changes. As stated by Joslin and Poole, "the simulation will be unrealistic if the plan is static" [1].

Human allocation uncertainty is seen as an example of a series of factors that can lead to a plan change. During the project run, based on the specific situations a team could be assigned a task that was originally assigned to another team if the latter one is unavailable (and the former team is available). The distribution function used to reflect the duration of the task most probably will be different to match the properties of the new task executers. Usually, teams have different levels of expertise which will in turn affect durations of tasks they perform. A practical simulation model should handle this dynamic aspect of a plan and anticipate possible changes.

In this paper we move one step towards the more realistic modeling of project schedules by incorporating a degree of anticipation of team allocation variability during the project execution. This uncertainty is modeled by means of human resource availabilities that are subject to unexpected changes.

We allow both duration and sequence of tasks to be variable – based on available resources and depending on various on-the-fly decisions by participants on the project. We simulate project schedules both with and without the possible on-the-fly decision scenarios. The objective of this is to study the effects of such changes in the project schedule and show their significance. Furthermore, we aim to provide an approach for accurate prediction of project schedule duration given the afore-described circumstances.

We use Gantt charts and state-transition diagrams for modeling project schedules. We extend both formalisms to model what we term as state-dependent (floating) task. Floating task represents a task that models uncertainty in human resources allocation and is a subject to various changes. We chose the proxel-based method for simulating the project schedules, as it has been successfully applied to this area [9, 10] and can provide highly accurate complete results.

Input probability distribution functions can be fitted based on historical data for similar tasks and situations and may be adapted to concrete situations of projects. The estimation process would, obviously, require a high level of expertise.

The remainder of this paper is structured as follows: In the next section we provide an overview of problems faced in the project scheduling simulation and we describe the proxel-based method. Further, we introduce the floating task model and our running example as a software development process. Then, we illustrate our simulation scenarios as well as our simulation results followed by a discussion on the importance of uncertainty and on-the-fly decision modeling in simulation model. Finally, we conclude.

2 Problem Definition

A project consists of a number of tasks (activities) where a predefined set of tasks has to be processed in order to complete the project. The tasks are in fact related by two constraints:

1. Precedence constraints: usually in a project development tasks cannot be undertaken in any order and some tasks cannot start unless others have been already completed; and
2. Resource sharing: performing tasks requires efficient resources' management. Such resources may include financial resources, inventory, human skills, production resources, information technology (IT), etc.

The incorporation of *uncertainty* into project planning and scheduling has resulted in numerous research efforts, particularly focusing on uncertainty in task duration or cost [4]. In our case, we are interested in studying the effect of human uncertainty factor on the duration of a project, in terms of on-the-fly decisions and resource allocation.

As running example, we consider a software development project. Typical requirements descriptions might include the task lists for people, and allocation schedules for resources.

Workforce allocation is seen as an important step in any software project management. It is the phase where all relevant elements of the software development process are taken into consideration for allocating software developers to the different project tasks [11].

While an initial allocation of software developers is calculated based on initial requirements, it is frequent that workforce adjustments during project performance becomes necessary for several reasons : (1) projections recalculation, based on the current workforce size, and the current development productivity [11], (2) number of remaining requirements to be implemented, and (3) requirements volatility.

Because of the above-mentioned reasons, it may happen that a team is assigned to a task that was originally assigned to another team during the workforce adjustment. Such scenario is in fact not easy to consider during the project scheduling. First, it is difficult to know when such adjustment will happen. This decision will be taken on-the-fly. Second, and more importantly, different teams have distinct expertise. Put in other words, the time that would take team A and team B to finish a task is not necessarily the same. It is frequent that one of the teams may need additional time to acquire the necessary expertise to achieve that particular task. When predicting the duration of project schedules, such scenarios should be considered.

The objective of our approach is to compute the probability distribution function of the duration of the project schedule taking into consideration human resource allocation uncertainty and typical on-the-fly decision behaviors. In addition we observe the effect that these behaviors might have on the duration on the project and want to stress the importance of their consideration.

3 The Proxel-Based Method

The proxel-based method [12, 13] is a relatively novel simulation method, whose underlying stochastic process is a discrete-time Markov chain [14] and implements the method of supplementary variables [15]. The method, however, is not limited to Markovian models. On the opposite, it allows for a general class of stochastic models to be analyzed regardless of the involved probability distribution functions. In other words, the proxel-based method combines the accuracy of numerical methods with the modeling power of discrete-event simulation.

The proxel-based method is based on expanding the definition of a state by including additional parameters which trace the relevant quantities in one model through a previously chosen time step. Typically this includes, but is not limited to, age intensities of the relevant transitions. The expansion implies that all parameters pertinent for calculating probabilities for future development of a model are identified and included in the state definition of the model.

Proxels (stands for probability elements), as basic computational units of the algorithm, follow dynamically all possible expansions of one model. The state-space of the model is built on-the-fly, as illustrated in Figure 1, by observing every possible transiting state and assigning a probability value to it (*Pr* in the figure stands for the probability value of the proxel). Basically, the state space is built by observing all possible options of what can happen at the next time step. The first option is for the model to transit to another discrete state in the next time step, according to the associated transitions. The second option is that the model stays in the same discrete state, which results in a new proxel too. Zero-probability states are not stored and, as a result, no further investigated. This implies that only the truly reachable (i.e. tangible) states of the model are stored and consequently expanded. At the end of a proxel-based simulation run, a transient solution is obtained which outlines the probability of every state at every point in time, as discretized through the chosen size of the time step. It is important to notice that one source of error of the proxel-based method comes from the assumption that the model makes at most one state change within one time step. This error is elaborated in [13].

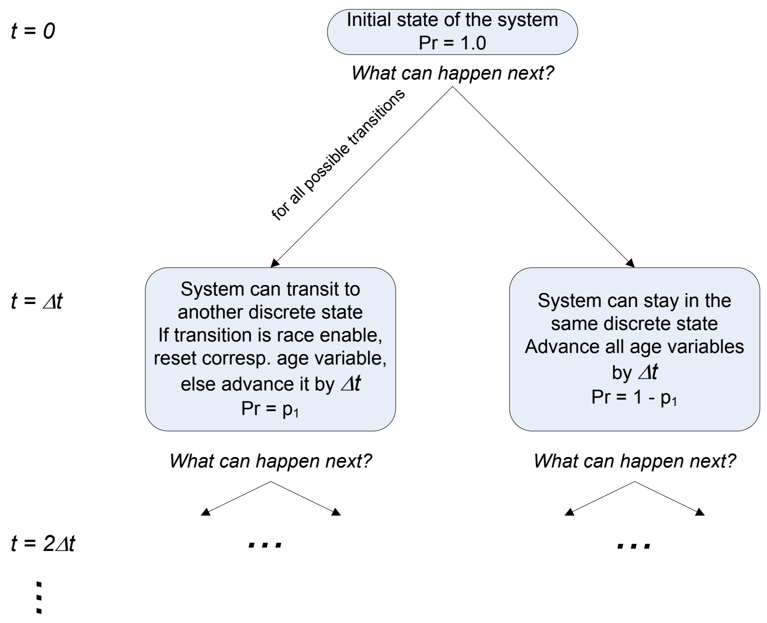

t = 0
Initial state of the system
Pr = 1.0
What can happen next?
for all possible transitions
t = Δt
System can transit to another discrete state
If transition is race enable, reset corresp. age variable, else advance it by Δt
Pr = p1
System can stay in the same discrete state
Advance all age variables by Δt
Pr = 1 - p1
What can happen next?
What can happen next?
t = 2Δt

Fig. 1. Illustration of the development of the proxel-based simulation algorithm

Each proxel carries the probability of the state that it describes. Probabilities are calculated using the instantaneous rate function (IRF), also known as hazard rate function. The IRF approximates the probability that an event will happen within a predetermined elementary time step, given that it has been pending for a certain amount of time τ (indicated as 'age intensity'). It is calculated from the probability density function (*f*) and the cumulative distribution function (*F*) using the following formula:

$$\mu(\tau) = \frac{f(\tau)}{1 - F(\tau)} \tag{1}$$

As all state-space based methods, this method also suffers from the state-space explosion problem [16], but it can be predicted and controlled by calculating the lifetimes of discrete states in the model. In addition, its efficiency and accuracy can be further improved by employing discrete phases and extrapolation of solutions [17]. More on the proxel-based method can be found in [13].

4 State-dependent (floating) task

4.1 Vital vs. non-vital tasks

To formalize uncertainty we define a highly uncertain *state-dependent task*, for which we allow any relevant parameters (including history of the project) determine its duration. We term this type of task as *floating* task. Its duration probability distribution is a complex function that among other factors depends also on the team that performs the task (its previous training, number of participants, etc.). The floating task supports introducing human decision uncertainty factors in project scheduling.

We classify all tasks into two categories, i.e. vital and non-vital, depending on their importance for the success of the project and the risk strategy of the project, i.e.:

1) Vital tasks: these tasks are estimated as critical for the success of the project. They are assigned only to *experienced professionals* to reduce the risk of their failure. Consequently, vital tasks are assigned a single team responsible for their implementation (fixed resource allocation strategy).
2) Non-vital tasks: these tasks are estimated as secondary for the success of the project. Non-vital tasks can be assigned to more than one team. Any of the teams that become available can implement it in order to optimize the project duration and maximize resource utilization. Under certain circumstances, non-vital tasks can be cancelled as well. In general, non-vital tasks invite various on-the-fly decision scenarios and can be often modeled by *floating tasks*.

Whether a task is vital or non-vital can be determined from the project requirements.

If we consider our running example, it is well known in software requirements management that users and stakeholders establish priorities to the feature set. Typical priority levels are: *critical*, *important*, and *useful* [18]. When simulating project schedules, we propose to categorize the set of prioritized features as vital and non-vital tasks. It is obvious that a critical feature with high risk cannot be seen as non-vital task since non-vital task may be even cancelled while a useful feature can be seen as a non-vital task. Introducing feature risk level as factor to decide about the categorization of the different features into vital and not vital tasks is out of the scope of this paper. Such issues are seen as part of our future work.

4.2 Case Study: the HOme Lightening automation System (HOLIS)

The HOme Lightening automation System is a product to be marketed by Lumenations, a worldwide supplier of commercial lightening systems for use in professional theater and amateur stage production. HOLIS is a home lightening automation system that brings new lightening automation functionality with ease of use, comfort, and safety. It is intended to be used by homeowners' buildings and high-end homes. Details of the case study can be found in [18]. To simulate the schedule of the development of HOLIS, we select a subset of the different features that the system should implement. Table 1 summarizes the subset of HOLIS system features with their respective priorities and the effort needed to implement each of them. Effort has one of the three typical levels: *low*, *medium*, and *high*. Unsurprisingly, the effort needed to implement a task is team dependent. The values given in Table 1 represent an estimation of the effort needed to a trained team to implement the feature.

Table 1. Features of the HOLIS System

Features	**Priority**	**Effort**	**Vital vs. NonVital**
Feature 1: automatic timing settings for lights and so on.	Critical	Low	Vital
Feature 2: built-in security features (alarm, bells)	Critical	Medium	Vital
Feature 3: non-PC control unit	Critical	High	Vital
Feature 4: vacation settings	Important	Low	Vital
Feature 5: uses my own PC for programming	Important	High	Vital
Feature 6: close garage doors	Important	Low	Vital
Feature 7: automatically turn on closed lights when door opened	Useful	Low	Non-vital
Feature 8: interface to audio/video system	Useful	Medium	Non-vital

To simulate the project duration, we first need to determine vital and non-vital features. As shown in the last column of the table, the first six features are all considered vital. It is mainly because of their priorities (*Critical* and *Important*). The last two features are considered non-vital because they are useful features that the customers would like to have, only if possible.

4.3 HOLIS System: Teams Allocation

In order to simulate the HOLIS system, we determine the human resource allocation. We suppose that we have two teams working on the HOLIS project: Team A and team B. It is obvious that these two teams have different expertise, and each will fit better to a particular task. As explained previously, any software project is subject to workforce adjustment. During this case study, we anticipate such on-the-fly decisions and allow the expression of such possible decisions, while simulating the project duration. Table 2 illustrates the distribution of the features between the two available teams. It also shows the estimated effort that the assigned team needs to implement the features. All the vital tasks are assigned to a team and this team is not subject to change. However, the two non-vital features may be implemented by any of the available teams (A or B). The effort needed for these two teams to complete the task is not necessarily the same. As we can see, team A is more trained to implement Feature 7 than team B, while team B is more trained to implement Feature 8 than team A.

Table 2. HOLIS System: Teams Allocation

Features	Effort (for trained team)	Vital vs. NonVital	Assigned Team and Estimation of the Needed Effort
Feature 1	Low	Vital	Team A (Effort = *Low*)
Feature 2	Medium	Vital	Team A (Effort = *Medium*)
Feature 3	High	Vital	Team B (Effort = *High*)
Feature 4	Low	Vital	Team B (Effort = *Low*)
Feature 5	High	Vital	Team A and Team B (Effort = *High*)
Feature 6	Low	Vital	Team A and Team B (Effort = *Low*)
Feature 7	Low	Non vital	Team A (Effort = *Low*) or Team B (Effort = *Medium*)
Feature 8	Medium	Non vital	Team A (Effort = *High*) or Team B (Effort= *Medium*)

The conceptual model of the schedule is as shown in Fig. 2. This model clearly states that Feature 7 and Feature 8 will be implemented by any of the available teams, to the contrary of the remaining HOLIS system features. Thus, features 7 and 8 will be modeled using floating tasks.

To avoid risk, the vital features are scheduled first (in parallel), after what the teams spend a certain time on the non-vital features (7 and 8). If that exceeds a certain amount of time and takes too long, then both teams are transferred to features 5 and 6 that require combined work of both teams.

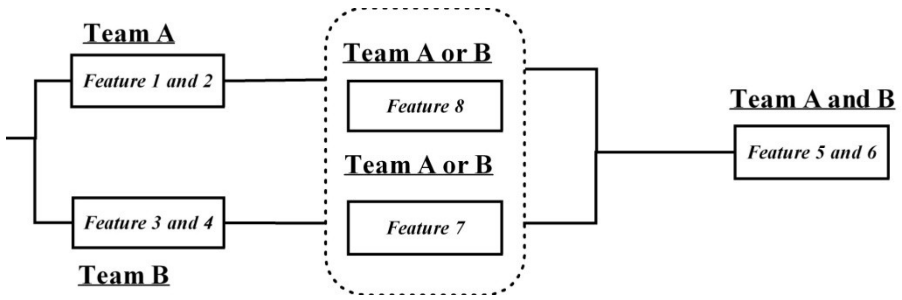


Fig. 2. Features Development Management

4.4 Simulation Model: Floating Task

In the following we present our sample model that we use to demonstrate our approach. The Gantt chart of the sample software development project schedule is shown in Fig. 3. Each of the tasks corresponds to a software requirement or combination of software requirements.

For simplicity reasons, we combine the features to represent tasks as follows:

- Features 1 and 2 into Task T1
- Features 3 and 4 into Task T2
- Feature 7 into Task T3
- Feature 8 into Task T4
- Features 5 and 6 into Task T5

thus resulting into a model with 5 tasks. Both, tasks T3 and T4 are floating tasks, and their processing depends on the state of the system including age intensities of tasks in progress.

The project schedule has two software developer teams assigned (A and B) and commences by running two tasks (T1 and T2, both vital) in parallel. Once either of the tasks is finished, the available team commences the task that is its favorable, i.e. lower effort (e.g. T3 for Team A, and T4 for Team B). When the team completes the task (T3 or T4), if the other team is still not finished on T1 or T2, then the former team starts working on the task that is rated as higher effort for it. If the originally assigned team completes finally its first task, and the other team has worked for a *short time* on the High Effort task, then it is cancelled, and both teams start working on Task T5. Alternatively, the available team waits for the other team to complete its task before both commence with T5.

Additionally, if the teams on the project are currently processing only non-vital tasks and it takes *too long*, they are interrupted, and both teams are assigned to start working on task T5.

In our sample model we observe two possible on-the-fly decision scenarios, as described in the following:

a) If the duration of task T3/T4 performed by team B/A after the team has completed its preferably assigned task (the one with lower effort) is “too short” then interrupt it and start processing task T5 by both teams.

b) If the duration of the project is taking “too long” and it is currently processing non-vital tasks, then all of them are cancelled and both teams start working on T5.

Apparently, there is fuzziness in the project schedule description (i.e. “too short”, “too long”) that requires adequate modeling. For that we use the following fuzzy function (note that there is no limit as to what function can be used):

$$g(x)=\begin{cases} 0, x<a \\ \dfrac{x-a}{b-a}, a\le x\le b. \\ 1, otherwise \end{cases} \tag{2}$$

The exact distribution functions that we use in our sample model are shown in Table 3. For simplicity reasons we have limited them to normal and uniform distributions, which is not a limitation of the approach. In addition, to describe the fuzzy behavior we have used the function shown by Equation (2), with the following parameters:

1) "too short", $1-g(x); a=0.0, b=5.0$
2) "too long", $g(x); a=10.0, b=15.0$

Both functions are illustrated in Table 4. In case (1) it means that the age intensity of the corresponding activity is less "too short" for 4.0 than e.g. for 2.0. Also, it means that it is extremely "too short" for the age intensity of 0.0, as well as it is not "too short" any more for the age intensity of 5.0. In case (2) the task has taken "too long" if its age is 15.0 at maximum, and the least "too long" if its age is 10.0.

Table 3. Input data for the sample project schedule (N-Normal, U-Uniform)

Features	**Vital vs. NonVital**	**Task**	**Assigned Team and Duration Probability Distributions**
Feature 1	Vital	T1	Team A: N(5.0, 1.0)
Feature 2	Vital		
Feature 3	Vital	T2	Team B: U(2.0, 10.0)
Feature 4	Vital		
Feature 5	Vital	T5	Team A and Team B: U(0.5, 2.0)
Feature 6	Vital		
Feature 7	Non vital	T3	Team A: U(2.0, 4.0)
			Team B: N(7.0, 1.2)
Feature 8	Non vital	T4	Team A: N(7.0, 1.0)
			Team B: U(2.0, 5.0)

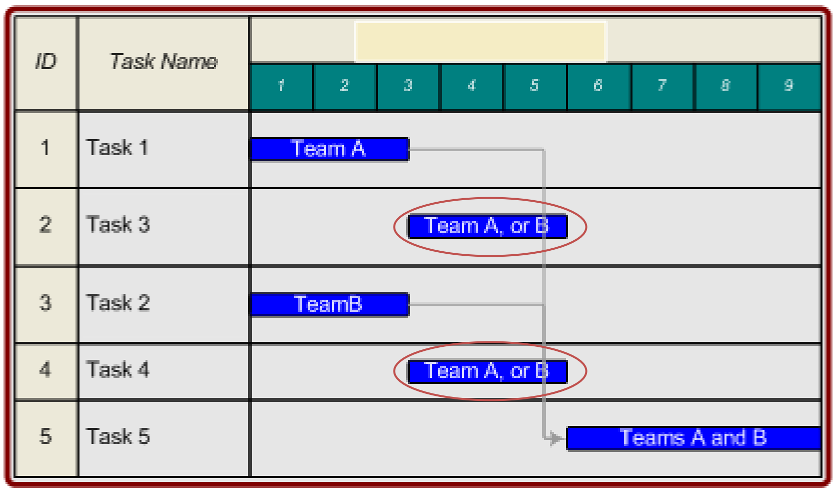
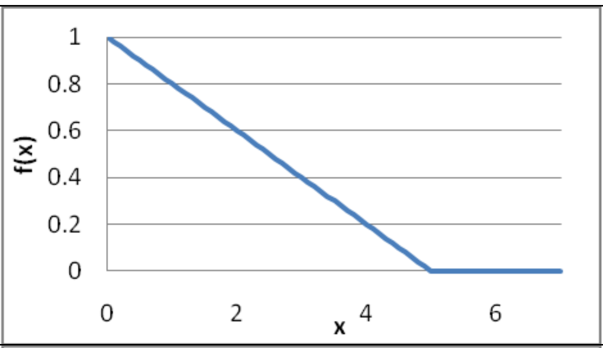
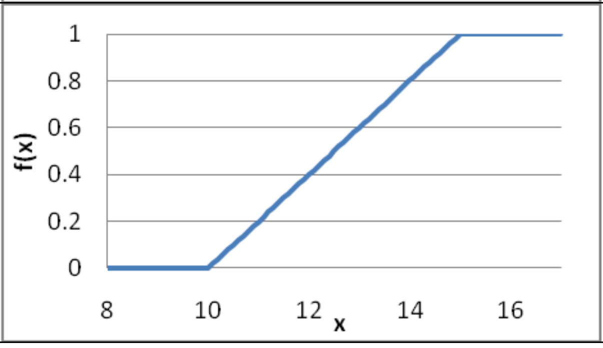


Fig. 3. Gantt chart of the example model, floating tasks are encircled in red color

Table 4. Fuzzy functions involved in the sample project schedule

"too short"	$a = 0.0,$ $b = 5.0$	
"too long"	$a = 10.0,$ $b = 15.0$	

5 Proxel-Based Simulation of Extended Project Schedules

In the following we provide the details of the proxel-based simulation of the sample project schedule that involves a floating task. This should serve as description of our approach through an example.

Each task in the model has a name, a priority level (vital or non-vital), duration probability distributions with respect to the possible team association, and a set of pre-requisite tasks. The proxel format of the state of the project schedule encompasses the following parameters:

- task vector $\{T_i\}$, where T_i is the task that team i is working on, or I for idle,
- age intensity vector $\{\tau_i\}$, for tracking the duration of tasks, and
- probability value.

Thus the format of the proxel is as follows:

Proxel = (*Task Vector, Age Intensity Vector, Probability*)

The initial proxel, i.e. the proxel that marks the initial state of the system would be ((T1, T2), (0, 0), 1.0). It describes the situation in which team A is working on task T1, and team B on task T2 with a probability of 1.0. In the next time step the model can do each of the following developments:

a) Task T1 completes,
b) Task T2 completes, or
c) None of the tasks completes.

Resulting into the following three proxels:

a) ((T3, T2), (0, Δt), p_1)
b) ((T1, T4), (Δt, 0), p_2)
c) ((T1, T2), (Δt, Δt), $1 - p_1 - p_2$)

In case (a), team A starts working on task T3, and also the corresponding age intensity is now reset to track the duration of T3. In case (b) team B takes over task T4. Case (c) shows the situation of both teams continuing what they have been doing before.

For demonstration, let us further develop the case (a). The next events that might happen are:

a1) Task T3 completes,
a2) Task T2 completes, or
a3) None of the tasks completes.

The interesting case is when T3 completes, which brings the model into state (T4, T2), where it subjected to various on-the-fly decision scenarios, as modeled by fuzzy functions. If team A has spent short time working on T4 (as it is considered non-vital), the task T4 is cancelled and both teams A and B proceed working on task T5. Else, team B waits for team A to complete task T4 before they proceed to task T5. For generating each new proxel, durations of tasks in progress need to be investigated for the decision modeling, as typically they are parameters of the fuzzy functions.

The state-transition diagram of the sample project schedule is shown in Fig. 4. As depicted with the extra-wide arrow ⇗, when team A is working on T4 and team B on T2 (state (T4, T2)), the transition associated with the completion of T2 depends on the time that team A has already spent on working on task T4. If it was "too long" then team B will stay idle and wait for its completion. On the other hand, if team A has just started working on task T4, then it is interrupted and both teams start working on task T5 which leads to completing the project. The same situation applies to the discrete state (T1, T3).

The red arrows in Fig. 4 shows the transitions due to Scenario (b), i.e. when the project currently processes non-vital tasks and its duration is too long. In that case there is a possibility that the tasks are interrupted and the project proceeds to task T5.

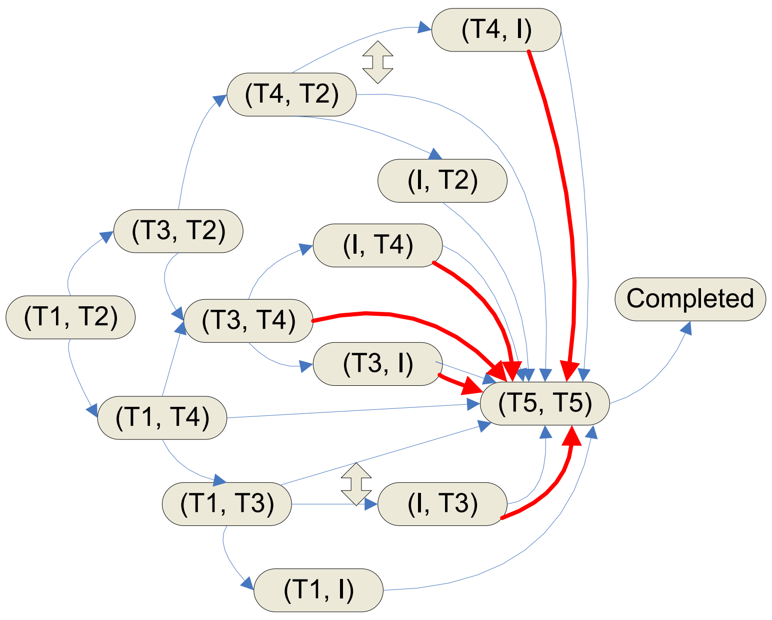


Fig. 4. State-transition diagram of the project schedule

The algorithm that we have developed represents an extension of the original proxel-based method [12, 13]. In particular, the differences can be summarized as: *Prior to processing each transition, check all possibilities for possible flow changes based on proxel parameters and corresponding to model description. Generate subsequent proxels correspondingly.*

To illustrate the changes of the original algorithm, we illustrate the scenario of development from the proxel ((T4, T2), (t1, t2), p) in Fig. 5 using a proxel sub-tree to represent the specific elements. It is evident that in the probability calculation of the proxel, the fuzzy function $g(t)$ participates too. It increases the branching of possible paths that the model can take, as denoted by the green colored proxels.

Calculating probabilities of proxels need to include the fuzzy functions (here $g(\tau)$) associated with the possible model developments, as follows:

$$probability = \mu(\tau_1) * g(\tau_2) \tag{3}$$

and correspondingly for the alternative development:

$$\mathit{probability} = \mu(\tau_1) * (1 - g(\tau_2)). \tag{4}$$

This implies that if the fuzzy function $g(\tau)$ is defined on (a, b), then during the time that the age intensity τ_2 is within this interval, the number of possible developments of the model grows, and includes the ones modeled by the fuzzy function. This means that the state-space of the model is dynamically changing with respect to the state parameters.

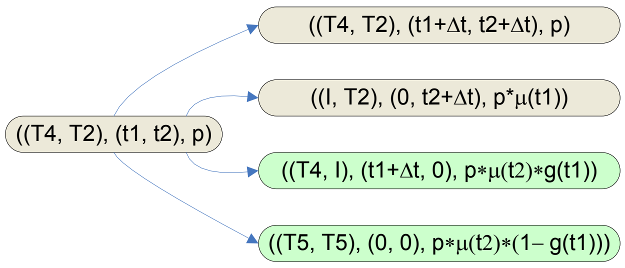

((T4, T2), (t1, t2), p)
((T4, T2), (t1+Δt, t2+Δt), p)
((I, T2), (0, t2+Δt), p*μ(t1))
((T4, I), (t1+Δt, 0), p∗μ(t2)∗g(t1))
((T5, T5), (0, 0), p∗μ(t2)∗(1− g(t1)))

Fig. 5. Proxel sub-tree developed for the proxel ((T4, T2), (t1, t2), p)

The initial proxel development is shown in Fig. 6. This represents the verbose output of our program and it shows how the program works. As it displays only the first 5 steps and the time step $\Delta t = 0.1$ there are only two discrete states that the model can be in: (T1, T2) and (T3, T2). This is so because the completion of task T2 is uniformly distributed from 2.0 to 10.0, thus it cannot transit to (T1, T4) before the 20th time step.

```
INITIAL PROXEL
AddProxel  ((T1, T2, ( 0dt,  0dt)),  1.00000e+000)

STEP 1
AddProxel  ((T3, T2, ( 0dt,  1dt)),  1.48672e-007)
AddProxel  ((T1, T2, ( 1dt,  1dt)),  1.00000e+000)

STEP 2
AddProxel  ((T3, T2, ( 0dt,  2dt)),  2.43896e-007)
AddProxel  ((T1, T2, ( 2dt,  2dt)),  1.00000e+000)
AddProxel  ((T3, T2, ( 1dt,  2dt)),  1.48672e-007)

STEP 3
AddProxel  ((T3, T2, ( 2dt,  3dt)),  1.48672e-007)
AddProxel  ((T3, T2, ( 0dt,  3dt)),  3.96130e-007)
AddProxel  ((T1, T2, ( 3dt,  3dt)),  9.99999e-001)
AddProxel  ((T3, T2, ( 1dt,  3dt)),  2.43896e-007)

STEP 4
AddProxel  ((T3, T2, ( 0dt,  4dt)),  6.36983e-007)
AddProxel  ((T1, T2, ( 4dt,  4dt)),  9.99999e-001)
AddProxel  ((T3, T2, ( 2dt,  4dt)),  2.43896e-007)
AddProxel  ((T3, T2, ( 1dt,  4dt)),  3.96130e-007)
AddProxel  ((T3, T2, ( 3dt,  4dt)),  1.48672e-007)

STEP 5
AddProxel  ((T3, T2, ( 2dt,  5dt)),  3.96130e-007)
AddProxel  ((T3, T2, ( 4dt,  5dt)),  1.48672e-007)
AddProxel  ((T3, T2, ( 3dt,  5dt)),  2.43896e-007)
AddProxel  ((T3, T2, ( 0dt,  5dt)),  1.01409e-006)
AddProxel  ((T1, T2, ( 5dt,  5dt)),  9.99998e-001)
AddProxel  ((T3, T2, ( 1dt,  5dt)),  6.36983e-007)
```

Fig. 6. Output of the proxel-based simulation for the first 5 steps

6 Experiments and Results

6.1 Experimental Environment

The experiments were run on a standard workstation with an Intel Core2Duo Processor at 2.0 GHz and 1 GB RAM. The choice for Δt was 0.1 and the simulation was run up to time $t = 25$. This implies that the number of simulation steps was 200.

The computation time for this experiment was ca. 5 seconds. In the following we present the results, i.e. the statistics that were calculated during this simulation experiment. The input data is provided in Table 3.

6.2. Experiments

The goal of the experiments is to show the importance of modeling the effects of on-the-fly human decision behaviors on project schedules. For that purpose we first simulated the project schedule in an ideal scenario, i.e. excluding any intrusions during project running. Next, we simulated the project duration exposed to the hypothetical scenarios (both (a) and (b)) for on-the-fly project flow decisions. To study the effect of neglecting them, we compare both solutions and present the results with a chart.

In Fig. 7, we observe the probability distribution of project duration for all combination of on-the-fly decision scenarios. It is evident that in the ideal case, i.e. the case that is not a subject to various on-the-fly decisions about the workflow, the project duration is the longest, as expected. However it is very interesting that the difference in duration is more than 5 time units (in this case more than 25% of the duration of the project). In the ideal case the project completes with a probability of 1.0 in ca. 20 time units. In the case with both scenarios this duration is ca. 13 time units, and in the case with only scenario (a) it is ca 18. Therefore, the difference is significant.

In order to represent closely the effect of the modeling and simulation of the on-the-fly workflow decisions, Fig. 8 shows the difference of the two probability distributions, with and without on-the-fly decision scenarios. The results show that in our sample model, it goes up to ca. 0.6, which is a very significant probability difference.

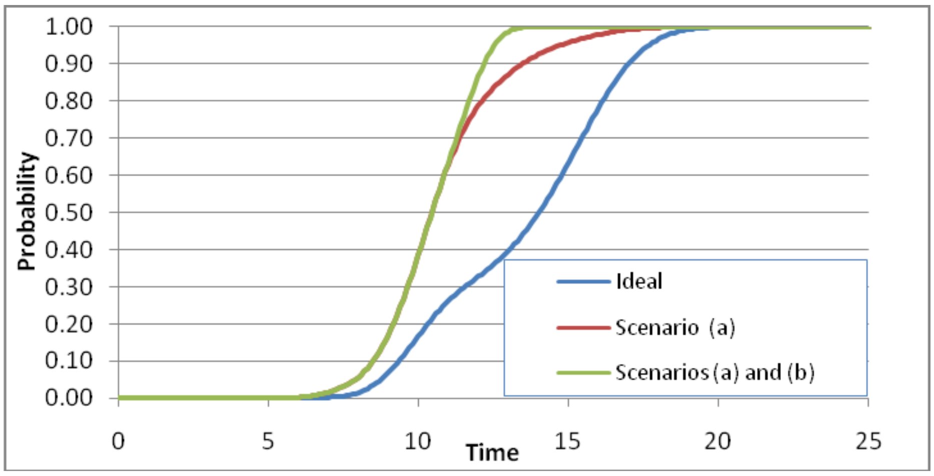


Fig. 7. Probability distribution of the duration of the project schedule with the 3 possible combinations of scenarios

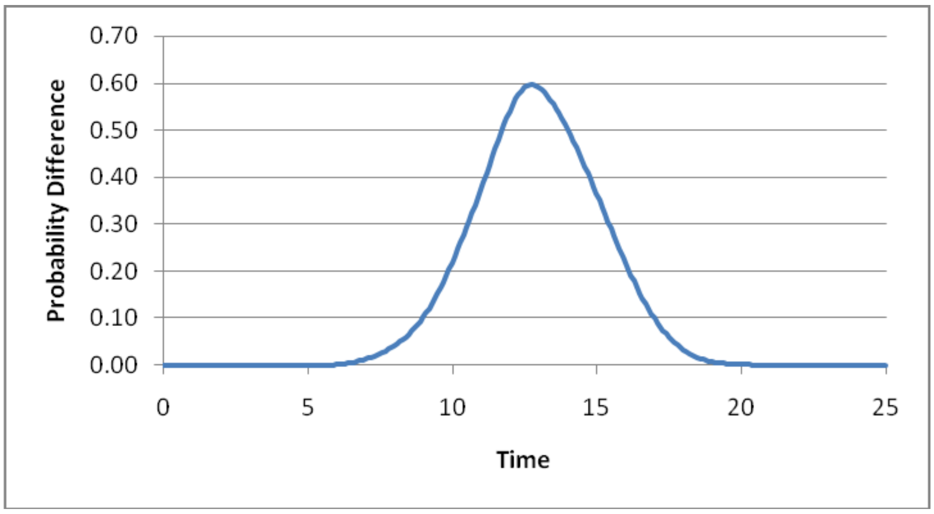


Fig. 8. Project schedule duration probability distributions difference for the ideal and project schedule with both scenarios

7 Discussion

The approach that we presented allows a higher degree of uncertainty in project schedules to be modeled and simulated. The uncertainty that we observe is in terms of duration of tasks, task allocation, as well as arbitrary on-the-fly decisions that influence the workflow. We all witness that these things happen almost every time and in every project. Thus, simulation models need to consider them in order to obtain accurate measures for the duration of project schedules. Very often, these factors are neglected, and by our example we showed what difference they can make.

In our example model, the probability difference for the completion of the project reached ca. 0.6, and this is still just a toy model. In real project schedules it can be more extreme and thus it has to be taken into account. The question that arises is how to obtain the numbers that represent and model these behaviors. We believe that they can be modeled by historical data and tracking from previous projects of similar types. In addition, expert knowledge and common sense can help to a great extent.

8 Summary and Outlook

This paper presents a more realistic project schedule simulation and modeling approach that allows for a high level of uncertainty. The purpose of this simulation model is: (a) to model the uncertainty of human resources allocation to the different project tasks and (b) to take advantage of this uncertainty to simulate various on-the-fly human decisions and study their impact on the project duration.

To extend our work we plan to address the effect of these uncertainty factors on the productivity and budget, by adding value, effort and cost parameters. In addition, we intend to extend our simulation model to handle the effects of requirements volatility in software engineering.

References

1. Joslin, D. and W. Poole. Agent-based simulation for software project planning. in Proceedings of the 37th conference on Winter simulation 2005. Orlando, Florida
2. Jing-wen, Z. and S. Hui-fang. Multi-Mode Double Resource-Constrained Time/Cost Trade-Offs Project Scheduling Problems. in International Conference on Management and Service Science, 2009. MASS '09. . 2009.
3. Huang, W., et al. Project Scheduling Problem for Software Development with Random Fuzzy Activity Duration Times. 2009: Springer.
4. Herroelen, W. and R. Leus, Project scheduling under uncertainty: Survey and research potentials. European Journal of Operational Research, 2005. **165**(2): p. 289-306.
5. Arauzo, J.A., et al., Multi-agent technology for scheduling and control projects in multi-project environments. An Auction based approach. Inteligencia Artificial, 2009. **42**: p. 12-20.
6. Sobel, M.J., J.G. Szmerekovsky, and V. Tilson, Scheduling projects with stochastic activity duration to maximize expected net present value. European Journal of Operational Research, 2009. **198**(3): p. 697-705.
7. AbouRizk, S.M. and R.J. Wales, Combined discrete-event/continuous simulation for project planning. Journal of Construction Engineering and Management, 1997. **123**(1): p. 11-20.
8. Perminova, O., M. Gustafsson, and K. Wikström, Defining uncertainty in projects–a new perspective. International Journal of Project Management, 2008. **26**(1): p. 73-79.
9. Rauch-Gebbensleben, B., K. Dammasch, and G. Horton, Generierung und Visualisierung des Zielkorridors in Projektplänen mit stochastischen Einflussgrößen, in Simulation und Visualisierung 2008. 2008: Magdeburg.
10. Isensee, C. and G. Horton. Proxel-Based Simulation of Project Schedules. in European Simulation Multiconference. 2004.
11. Pfahl, D. and K. Lebsanft, Using simulation to analyse the impact of software requirement volatility on project performance. Information and Software Technology, 2000. **42**(14): p. 1001-1008.
12. Horton, G., A new paradigm for the numerical simulation of stochastic Petri nets with general firing times. Proceedings of the European Simulation Symposium, 2002.
13. Lazarova-Molnar, S., The Proxel-Based Method: Formalisation, Analysis and Applications, in Faculty of Informatics. 2005, University of Magdeburg: Magdeburg.
14. Stewart, W.J., Introduction to the Numerical Solution of Markov Chains. 1994: Princeton University Press.
15. Cox, D.R., The analysis of non-Markovian stochastic processes by the inclusion of supplementary variables. Proceedings of the Cambridge Philosophical Society, 1955. **51**(3): p. 433-441.
16. Lin, F.J., P.M. Chu, and M.T. Liu, Protocol verification using reachability analysis: the state space explosion problem and relief strategies. ACM SIGCOMM Computer Communication Review, 1987. **17**(5): p. 126-135.
17. Isensee, C. and G. Horton. Proxel-Based Simulation of Project Schedules. 2004: Citeseer.
18. Leffingwell, D. and D. Widrig, Managing Software Requirements: a use case approach. 2003: Pearson Education.