Simulation-based Feature Selection for Software Requirements Baseline

Rabeb Mizouni
Department of Software Engineering, Khalifa University
Abu Dhabi, United Arab Emirates
rabeb.mizouni@kustar.ac.ae

Sanja Lazarova-Molnar
Faculty of Information Technology, United Arab Emirates University, PO Box 17551,
Al Ain, United Arab Emirates
sanja@uaeu.ac.ae

Abstract—**Requirements baseline is the set of features intended to be delivered in a specific version of a software application under development. During this decade the constant growth of software products along with the evident pressure on time to market has made the selection of features a crucial step for a software project success. It is both a challenging and time consuming process that requires a substantial expertise from project managers. Prioritization of features is one of the means that help in making the choice. It is typically performed by grouping features into three priority levels: *critical, important,* and *useful*. Critical and important features are seen as "must have", while useful features are qualified as "nice-to-have". Paradoxically, the latter plays an important role in customer satisfaction and achieving the "wow" factor. A good selection of useful features identifies efficiently those features that can be delivered by the end of the project without any additional delay. So far, managers have little support in this process increasing the chances of making a poor selection. To answer this need, we propose a new modeling and simulation approach that takes into account feature priorities and calculates the probabilities of having useful features implemented within the timeframe of the project. It also incorporates uncertainties related to human resources availability providing a more realistic schedule and estimation.**

Index Terms— **Requirements Baseline, Feature Selection, Features Priority, Simulation, Proxel-based Simulation**

I. INTRODUCTION

Project management is the discipline of planning, organizing, and managing resources to achieve specific project goals and objectives. It is the activity which uses schedules to plan and subsequently report progress within the project environment. In the initial project stages, project managers are usually concerned with defining initial project plans and requirements baseline where the project mission as well as the project schedule plan is identified [1]. This initial plan is used as a basis on which delivery commitments are made. Therefore, constructing credible initial plans that provide a good estimation of completion dates helps the project success and may avoid customers' disappointment.

A. Problem Statement

With the ever-growing size of software projects, managers are facing the growing problem of feature selection. Product feature is defined as a set of logically related requirements that provide certain functionality to the software and enable the fulfillment of business objectives. Thus, feature selection addresses the problem of selecting features that can be implemented within project constraints, such as human resources, budget, and time. Typically, the set of features that should be implemented in each release are identified in the initial plan. However, this process is not yet controlled and even though feature selection is carried out based on the available project resources, in many cases, managers fail to deliver all the features they first promised to their clients. In fact, activities in a real industrial project may take more time than their original estimate, leading to a delay in other activities due to resource unavailability [2] which may cause delays in features' implementation. Consequently, even good projects fail [3], particularly so in the software industry where statistics show that approximately one-third of software projects fail to deliver anything, and another third deliver something workable but not satisfactory [4]. This failure is mainly to poor upfront planning, late re-planning and non-tracked planning.

"Planning and control" has been named as one of the top three factors that influence project success, besides "project objectives" and "personnel and team building" [5]. In addition, it is well recognized today that the quality and depth of early planning is a common element on most successful projects. Such plans should remain of high quality even when the environment deviates due to uncertainties [6]. They also define accurately the product features to deliver within the timeframe of the project and which become the initial baseline for product design.

Requirements baseline is the set of features intended to be delivered in a specific version of the application. This baseline represents a contract between the customer and the development team. It reflects, in customer's point of

doi:10.4304/jsw.7.7.1440-1450

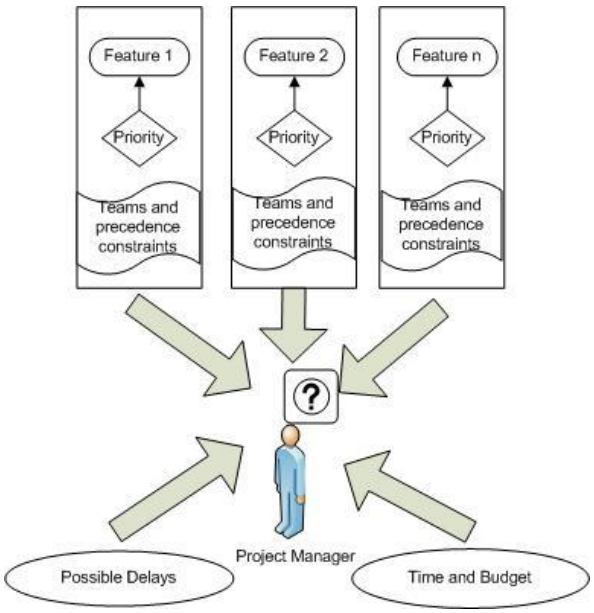

Figure 1: Traditional Approaches for Project Management : Manager Challenge

view, a specification of acceptable product to be delivered during future releases, as well as, in development team's point of view, the set of features that have reasonable probability of achievement [7].

A common technique used to help the definition of requirements baseline is prioritization. Prioritization relies on giving more weight to the most important features. These priorities are usually set by customers and attributed to features according to customer' needs. A common approach to prioritization is to group features into three priority categories [8]: *critical*, *important*, and *useful*. While *critical* and *important* features have to be delivered to the customer by the end of the project, *useful* features are seen as *nice-to-have* requirements that are not crucial for the success of the project. This prioritization helps the project manager to resolve conflicts and, more importantly, to perform trade-offs throughout the development lifecycle [9].

Based on above-mentioned features' priorities, we notice that there exists a fundamental difference in the nature of features. Project managers have certain flexibility in their choices among these *useful* features with respect to resources availability and deadlines, a flexibility they certainly do not have with *critical* and *important* features. However, with the hard competition that software industry is witnessing, implementation of *useful* features is playing an important role in the acceptance of the project and the "wow" factors that any development team is looking for. Unfortunately, due to unpredictable events and delays, managers are often forced to cancel the implementation of some nice-to-have features to meet releases' deadlines. While this seems the least harmful decision to cope with uncertainties, it still may affect badly the customer satisfaction. To avoid such situations, managers have to deliver a quality project baseline. Improving project baseline is in fact predicting in a more precise way the set of features that will be implemented even if the project plan deviates.

B. Contributions

To support the useful feature selection process, managers need an intelligent support for software release planning. Such support will:

1) help managers cope with the increasing number of features in software projects on the one hand, and with uncertainties that the project may face, on the other hand.

2) make the decisions made in requirements management not solely based on human judgment, as it is the case in most traditional approaches (see Figure 1), but also based on a sound model and methodologies.

3) be an intelligent support that reflects features' priority as well as uncertainties that surround any software in project scheduling. Last but not least,

4) help the optimizaion of resources' usage since projects should not be delayed or go over budget because of the implementation of these features.

In this paper we propose a new simulation approach that helps managers in the selection of useful features, as shown in Figure 2. Our approach promotes a proactive scheduling that takes into consideration the available human resources to schedule task's execution with respect to the priority of the feature it is implementing. Analogous to features' priorities, our model distinguishes between cancelable and non-cancelable tasks.

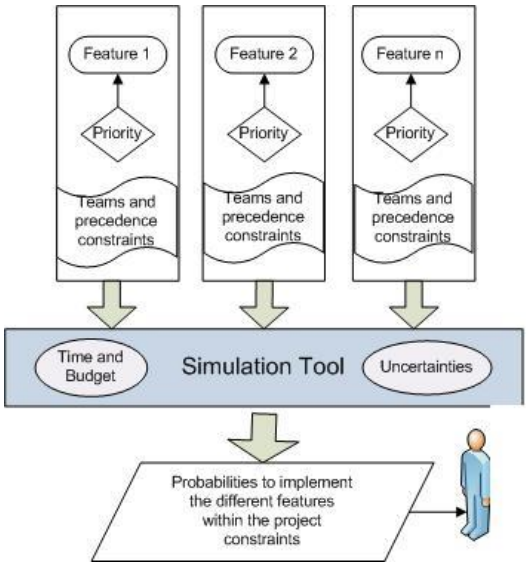

Figure 2: New Simulation approach to help feature selection

A cancelable task is defined as a project task that can be postponed or canceled, i.e.

- *postponed,* if:
  1) resources needed to implement the task are not available, or

2) there is a task with higher priority that is not implemented yet and that needs the resources of the cancelable task, and

- *canceled,* if:

3) the project runs out of budget and time.

To achieve our goal, we enhance the project schedule model to enable calculation of probabilities of having a certain task implemented by a given user deadline. Once the simulation results are available, probabilities of completing each cancelable task are provided to the manager. These probabilities may act as a new selection criterion and help them in making a judicious choice of features to consider in their baseline. The obtained baseline will have a higher probability to be implemented and delivered within the given deadline.

We use Gantt chart for modeling initial project schedules to display the precedence constraints. We extend this formalism to include cancelable tasks and to calculate the probability of their completion. We choose the proxel-based method for simulating project schedules for its flexibility and accuracy [10]. Namely, with one single simulation run, the hybrid method provides complete transient solution of a stochastic model, showing its behavior at every point in the discretized timeline. The method has been successfully applied to the simulation of classical project schedules [11].

C. Paper Organization

The rest of the paper is organized as follows. Section II presents the related work on feature selection as well as on simulation models in project scheduling. Section III defines different project tasks according to feature priority levels. It also presents the proxel-based simulation. Section IV illustrates our approach with a simulation example that shows how our model can help the selection of useful features. Section V concludes the paper and outlines our future work.

II. RELATED WORK

Unfortunately, major project planning software packages are too stiff when it comes to defining project schedules. Also, many of the analysis methods and tools oversimplify the uncertainty in projects and thus provide inaccurate results [12] [13, 14].

In [16], authors present a new framework, *NextMove*, that assists project managers in allocating and managing tasks in an agile, distributed development environment. The framework considers team backlog, as well as requirement priorities to help project teams in tracking, coordinating and communicating tasks in a distributed development environment. However the simulation model used does not consider emergent uncertainties that any software project often encounters. To have a realistic definition and analysis, project schedules have to anticipate high uncertainty, and should also provide recommendations to aid the decision making process in various possible uncertain scenarios. This is what we term an Enhanced Project Schedule, for which generation we have already developed a framework [15].

On the other hand, because of the importance of release planning, many researchers have addressed the assignment of requirements to a sequence of releases. The authors in [17] analyze the effects of defect and effort re-estimation in the process of release re-planning. Each planned release has a limited effort capacity. This capacity limits the number of features that can be implemented during that release. In the example they present, the features of the baseline were chosen according to their respective priorities. However, uncertainties that may arise are again not taken into account, affecting badly the feature selection. In [18], the authors present a six step process model for release planning, termed EVOLVE. This approach takes into account stakeholder priorities as well as effort constraints for all releases. While their approach supports the feature selection decision according to their priorities, it does not include resource constraints and other aspects of task scheduling, a fact that impairs the accomplishment of a robust schedule.

In our previous work [19, 20], we aimed towards providing a more realistic model and more accurate predictions of durations of project schedules. We have developed a new type of activity in project schedules, termed “floating task”. Floating task is a task that anticipates high uncertainty and is highly flexible in terms of its human resource allocation. In addition, this task has the property of being flexible in its order of execution with respect to other tasks. In this paper, we extend our model to help managers decide on features selection and answer the following three challenges:

1) How can project scheduling differentiate between nice-to-have features and vital features?
2) How to take full advantage of the global resource pool and try to implement the maximum number of nice-to-have features within the timeframe of the project?
3) How can simulation-based tools guide the manager in her/his choice of nice-to-have features to deliver in each release?

III. PROJECT SCHEDULING: A NOVEL MODEL

To provide a more realistic modeling of software project schedules, we consider a multi-modal task representation where each activity can be processed in one of several modes. Each mode describes a task implementation option in terms of duration and human resource allocation. We introduce several types of project tasks to enable proper mapping of each feature priority to a task in the project schedule.

TABLE 1: MAPPING BETWEEN FEATURE PRIORITY AND TASK NATURE

	Non Cancelable with fixed human resource allocation	Non Cancelable with multimodal human resource allocation	Cancelable with fixed human resource allocation	Cancelable with multimodal human resource allocation
Critical	X	X		
Important	X	X		
Useful	X	X	X	X

A. Assumptions

In the following we list the set of assumptions that our model is based upon:

1. Managers are provided with a pool of human resources with different degrees of productivity to perform various types of tasks.
2. Each team is responsible for completing the implementation of a feature as a whole.
3. Tasks are sharing human resources and their implementation is a subject to a certain precedence order.
4. A change in the team structure is considered as a change of the whole team. The new team has new characteristics and hence may take less or more time to implement its assigned tasks.
5. Duration of a task is modeled by a probability distribution function. 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.
6. A feature can be implemented by various teams. Each team has a specified probability distribution to implement it according to its expertise.
7. Features are correctly prioritized. Change of prioritization [21], as it may happen during the project implementation due to addition/removal/change of other features, is not observed in our current model.
8. The type of a project task depends on the priority of the feature that the task is representing.

B. Tasks Definition

We propose to distinguish tasks so that they reflect the priority of the features they are implementing. Two factors are to be considered: 1) flexibility of the order of execution in the schedule, and 2) flexibility in the human resources allocation. Further, we distinguish two tasks, named: *non-cancelable* tasks and *cancelable* tasks defined as follows:

1) Non-Cancelable Tasks: tasks that may have flexibility in human resource allocation, but cannot be canceled. In fact, when teams responsible for implementing a non-cancelable task are either unavailable or solicited to do tasks with higher priority, then these tasks can be postponed but never canceled. Resource allocations for non-cancelable tasks can follow a fixed strategy, where only one team can be assigned to implement them, or a multi-modal strategy, where many teams can be assigned to implement them according to teams availability.

2) Cancelable Tasks: tasks that have flexibility in of the human resource allocation and can be canceled. When teams responsible for implementing a cancelable task are either unavailable or solicited to implement tasks with higher priority, and the project is overdue, then the task is canceled. Resource allocations for cancelable tasks can follow a fixed strategy, where only one team can be responsible for implementing it, or a multi-modal strategy, where many teams can be responsible to implement it with respect to their availability.

Table 1 presents a possible mapping between features priority levels and their potential assigned tasks. Let us outline some facts:

1. As expected, critical and important features can never be modeled as cancelable tasks. Only nice-to-have features can be canceled.
2. While it is possible to model useful features with non cancelable tasks, we do not recommend it. It prevents the model from certain flexibility in the task execution order.
3. Critical and important features have to be modeled as non-cancelable tasks, either with fixed or multimodal human resources' allocation. We believe that this depends on the risk level of the feature. When the risk associated with the feature is high then it would be more judicious to assign its implementation to an experienced team and model it as non-cancelable task with fixed resource allocation. However, when the feature is critical, but presents low risk, then we can tolerate more flexibility in its implementation.

Let us formally define a task.

Let $R = \{R_i, 1 \leq i \leq n\}$ be the set of n features of the software under implementation. Let $T = \{T_i, 1 \leq i \leq m\}$ be the set of m teams (representing the human resources available for that software project).

Definition 1: Task

A task is a 3-tuple $task = (R_i, D, type)$ where:

1) $R_i \in R$ the feature the task is implementing.
2) $type \in \{\boldsymbol{cancelable}, \boldsymbol{non-cancelable}\}$ the type of the task
3) $D = \{D_{ij} = (R_i, T_j)/T_j \in T\}$ is the set of probability functions assigned to team T_j to implement feature R_i.

In the case of fixed human resource allocation strategy D is defined as a singleton.

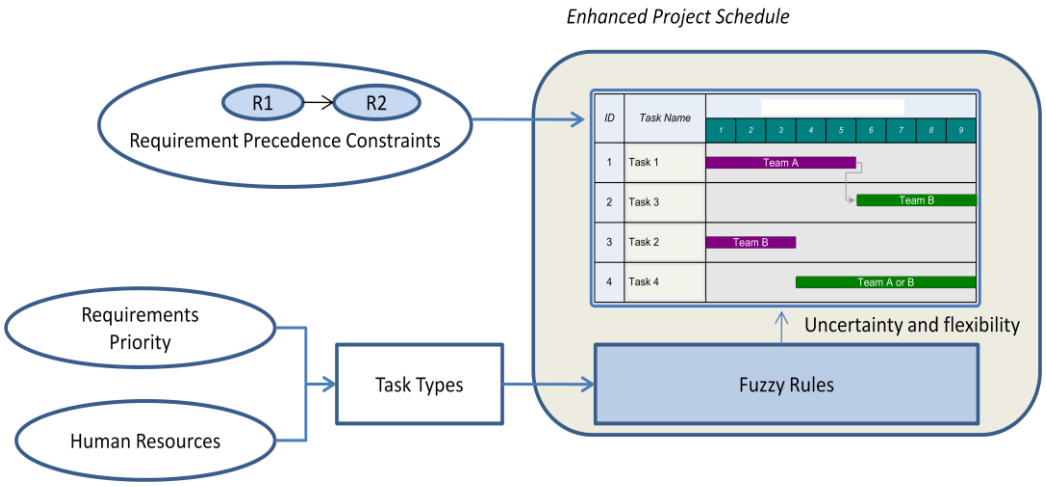

Figure 3: EPS Generation

C. Enhanced Project Schedule

Ideally, the nature of tasks needs to be taken into consideration when simulating the project schedule. To achieve this goal, we propose to extend the project schedule with *fuzzy rules* [22, 23]. In the context of our approach, fuzzy rules are conditional statements that express potential deviations to the initial project plan and remedial actions that should be undertaken consequently.

Each fuzzy rule is made up of two parts: *condition* and *action*, formally written as "condition ⟹ action". Conditions can be described either by using strict terms, or fuzzy ones. Actions can typically be canceling or interrupting some of the tasks, or one of the various types of rescheduling. Using these fuzzy rules makes our schedule description evolving, rather than rigid and inflexible. An example of a fuzzy rule would be:

$$Task_x\ takes\ too\ long \Rightarrow cancel\ Task_y$$

or

$$Task_x\ completes\ quickly\ after\ Task_y \Rightarrow cancel\ Task_z.$$

Both are examples for typical actions during project execution. However, in our approach we allow for their modeling, assessment and quantitative evaluation. The fuzzy rules are in fact the interpretation of the task type on the simulation schedule. They should be consistent with the type of the tasks specified by the analyst. As an example, a rule can never recommend to cancel a non-cancelable task as they represent critical and important features. Consequently, the two rules mentioned-above are correct only if Task z and Task y are cancelable tasks.

Definition 2: Enhanced Project Schedule

An Enhanced Project Schedule (EPS) is a 5-tuple $EPS = (Tasks, P, T, F, Initial)$ where:

1) $Tasks = \{Task_1, Task_2, \dots, Task_n\}$, set of tasks in the project schedule
2) $P = \{P_1, P_2, \dots, P_m\}$, where $P \subseteq Tasks \times Tasks$ is the set of tuples representing the tasks precedence constraints. The tuple $(Task_x, Task_y)$ would mean that completing $Task_x$ is a pre-requirement for beginning $Task_y$
3) $T = \{T_1, T_2, \dots, T_n\}$ set of teams available for the project implementation
4) $F = \{F_1, F_2, \dots, F_s\}$ set of fuzzy rules that are in line with features' priorities.
5) $Initial$ set of possible starting points of the project implementation.

Note that *Initial* represents the assignments of tasks to the different teams at the starting point of the project implementation. *Initial* can be either a singleton, where we have only one possible starting point to the project implementation or a set of different possible starting points. $Initial$ can also be a Gantt Chart that determines the initial assignment of tasks to available teams, as well as a possible initial order of execution.

As shown in Figure 3, the generation of EPS is based on the feature precedence constraints and the nature of the tasks. As mentioned previously, any rule specified within the fuzzy rules should not violate them.

D. Proxel-Based Simulation for Feature Selection

The proxel-based method is a simulation method based on the method of supplementary variables [24]. It was introduced and formalized in [10, 25]. The advantages of the proxel-based method are its flexibility to analyze stochastic models that can have complex dependencies and at the accuracy of results, which is comparable to the accuracy of Markov chain numerical solvers [26]. It has been successfully applied for project schedule simulation [11], and due to its flexibility it is highly suitable to model and simulate project schedules with additional

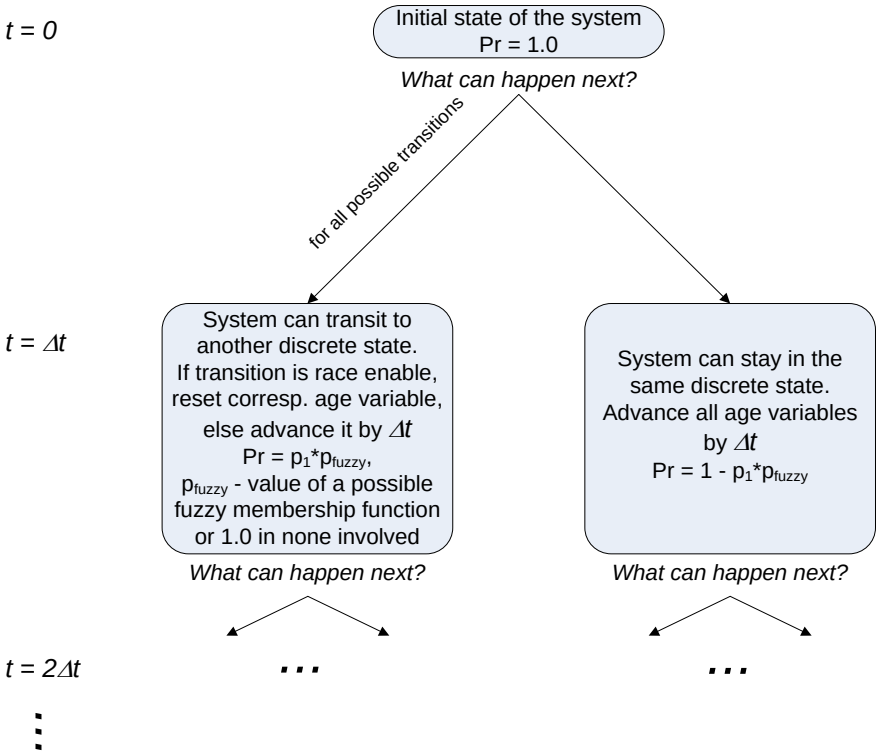

Figure 4: Illustration of the development of the proxel-based simulation algorithm

complexity of re-scheduling, governed by the inclusion of fuzzy rules.

The proxel-based method is based on expanding the definition of a state by including additional parameters which trace relevant quantities in one model following 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 a 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 4, 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 [10].

Each proxel carries the probability of the state that it describes (denoted as *Pr* in Figure 4). Probabilities are calculated using the instantaneous rate function (IRF), also known as hazard rate function. 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:

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

As all state-space based methods, this method also suffers from the state-space explosion problem [27], 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 [28]. More on the proxel-based method can be found in [10].

For our purpose we extended the original proxel-based simulation algorithm to account for the fuzzy scenarios (shown by the p_{fuzzy} variable in Figure 4). They fitted straightforwardly into the existing framework. In addition, the algorithm was adapted to collect statistics about the probability of having a certain feature implemented.

Algorithm 1: Proxel-based simulation of enhanced project schedules

```
Input: EPS, Project Goals
Output: Simulation Results
switch = 0
insert Initial State Proxel in the Proxel_Tree[switch]
switch = 1 - switch
 while (maximum simulation time has not been reached)
 {
   px = get_proxel(Proxel_Tree[switch]);
   for (each task in the Task Vector(px))
  {
     check task precedence & team availability;
     generate next state S;
      compute probability for S in computed_prob
      search for the S in the Proxel_Tree[1-switch];
      if (S found)
            {
              px1 = found_proxel(S);
              probability(px1) = (probability(px1) ) +
                  computed_prob;
            }
      else
           {
              generate new proxel px2(S);
              insert proxel in Proxel_Tree[1-switch];
            }
      delete px from Proxel_Tree[switch];
      increase simulation time by one time step;
      calculate statistics with respect to project goals;
      switch = 1- switch;
   }
}
```

The general simplified proxel format is the following:

$$Proxel = (State, t, Pr)$$

where:

$$State = (Task\ Vector, Age\ Vector, Completed\ Tasks), \text{ and}$$

- *Task Vector* is a vector whose size is equal to the number of teams available and records the task that each team is working on,
- *Age Vector* tracks the length that each team has been working on the task specified in the *Task Vector*, correspondingly,
- *Completed Tasks* stores the set of completed tasks,
- t is the time at which the afore-described state is observed, and
- Pr stores the probability that the schedule is in the afore-specified state at time t.

Algorithm 1 demonstrates the on-the-fly building of the state-space of the project schedule model. Thus, there is no need for any pre-processing to generate the state-space. It is directly derived from the input file specification. The initial state proxel is derived from the initial state that is specified in the input file as well.The algorithm operates by using two interchangeable data structures, Proxel_Tree[0] and Proxel_Tree[1], that store the proxels from two subsequent time steps (regulated by the *switch* variable). If two proxels represent the same state, there is only one proxel stored, and their corresponding probabilities are summed up.

For collecting statistics about useful features (cancelable tasks), we introduce rewards in the simulation model that are associated with the event of completion of a task. This provides us with a probabilistic assessment of the completion of a task, subject to the fuzzy scenarios associated with the project schedule. Finally, the obtained results are probability functions of time that show the probabilities of having each task completed. The ones that are most relevant for our approach are those of the cancelable tasks as they provide us with insight useful to the selection of features to be implemented within the timeframe of the project or the release. More precisely, we are looking for the *ranking* of probabilities of having each "nice-to-have" feature implemented.

Table 2: Features vs. Tasks and Human Resources Allocation

Features	Priority	Precedence Constraints	Human Resources Allocation	Task	Task Nature
Feature 1	Critical	Null	Team A	Task 1	Non-cancelable
Feature 2	Critical	Feature1	Team B	Task 2	Non-cancelable
Feature 3	Important	Null	Team B	Task 3	Non-cancelable
Feature 4	Useful	Feature3	Team A or B	Task 4	Cancelable
Feature 5	Useful	Null	Team B	Task 5	Cancelable
Feature 6	Useful	Null	Team A	Task 6	Cancelable

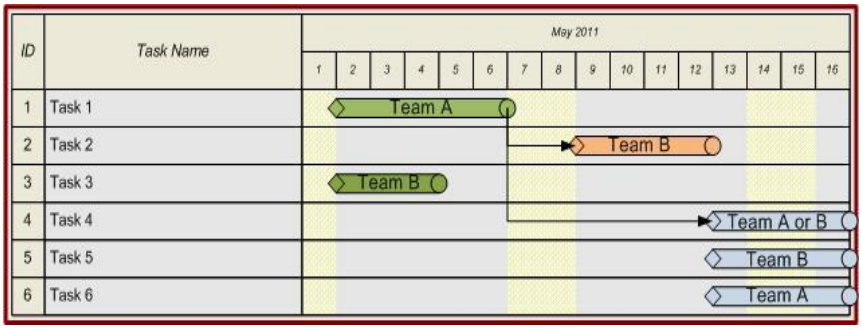

Figure 5: Initial Gantt chart (blue tasks are cancelable)

The adapted proxel-based simulation method is illustrated in Subsection IV.B, where using our example project schedule model, we show step-by-step the simulation process.

IV. EXPERIMENTS

We consider a general example of a project schedule that contains six features: two critical features, one important feature and three useful features. Based on their priorities, each feature is mapped to a task (Task1, Task2, Task3, Task4, Task5, and Task6) and assigned a task nature (i.e. cancelable or non-cancelable). We consider that the project has two teams available: *Team A* and *Team B*. Tasks 1, 2, 3, 5 and 6 have fixed human resource allocation while Task4 can be implemented by either *Team A* or *Team B*. Table 2 summarizes the mapping between the features and tasks as well as the human resources allocation. We notice that Feature2 and Feature4 cannot be implemented until Feature1 is completed, a fact that adds some precedence constraints to our model.

A. Model Specifications

The purpose of simulation is to help managers determine which of the three useful features have higher probability to be implemented within the project deadline, under the fuzzy rule constraints, and with respect to teams' availability, and hence be part of the project baseline. The Gantt chart of the sample project schedule is shown in Figure 5, where the blue-colored tasks are cancelable. In addition to this, the project schedule has a predefined deadline Δ=15.

The project schedule is described as an enhanced project schedule, which means it also features a fuzzy scenario, under which conditions it is observed. The fuzzy scenario enhances the degree of uncertainty available and creates a dynamically evolving project schedule, based on an initial one. In our case, the fuzzy scenario is defined as follows:

If the duration of the project is close to the deadline Δ, and all non-cancelable tasks are completed, then do not start any cancelable task in-line and do not interrupt the other team if they have already started to work on either of the tasks.

It is formally described as:

$$t \text{ is close to the } \Delta \Rightarrow \text{cancel uncommenced } Task_y, y \in \{4,5,6\}.$$

Recall that our goal is to discover, based on the available information, which of the "nice-to-have" features are most probable to be implemented, given the constraints of the initial project schedule. For this purpose we run proxel-based simulation which allows us to collect the necessary statistics to answer this question.

The concrete parameters of the tasks' duration distribution functions of our model are as follows:

- Duration of Task 1 ~ Uniform (2.0, 10.0)
- Duration of Task 2 ~ Normal (6.0, 1.0)
- Duration of Task 3 ~ Uniform (2.0, 6.0)
- Duration of Task 4, performed by:
  - Team A ~ Uniform (3.5, 5.5)
  - Team B ~ Uniform (2.0, 5.0)
- Duration of Task 5 ~ Uniform (0.5, 2.0)
- Duration of Task 6 ~ Uniform(0.2, 5.8)

The fuzzy membership function that describes *close to deadline* is defined as follows:

$$\lambda(t,a,b) = \begin{cases} 0, & t < a \\ \dfrac{t-a}{b-a}, & a \le t \le b \\ 1, & t > b \end{cases}, \text{ where}$$

$$a = \Delta - \frac{\Delta}{4}, b = \Delta$$

B. Simulation Details

To illustrate the proxel-based simulation in the context of enhanced project schedules we use our sample model and provide the step-by-step development of its simulation. Thus, the first step is to define the format of the proxel, which is dependent on the model to allow the tracking of the relevant quantities in the model. The proxel in the simplified form, by definition, consists of five

components, i.e. the state, the age intensities, the relevant rewards, the simulation time *t*, and the probability of the system being in that state at that point in time. In this way, it uniquely defines each and every state the model can be in every point in time.

For our model, the state definition is the following:

$$State = (TaskOfTeamA, TaskOfTeamB),$$

where both elements hold the names of the tasks that both teams are working on, correspondingly. Furthermore, we track the amount of time that each team has been working on the respective task, which forms the age intensity vector. Finally, to this we add the set of completed tasks, as an additional parameter (relevant reward) to the state vector that gets updated each time a task completes. Thus, the proxel definition for the example model is as follows:

$$Proxel = (State, AgeVector, CompletedTasks, t, Pr)$$

For the concrete example, the initial proxel is:

$$InitialProxel = ((Task1, Task3), (0,0), \emptyset, 0, 1.0)$$

At the beginning, teams A and B work on Task1 and Task3, correspondingly, and have been doing this for zero duration of time. There are no completed tasks, and thus, this parameter is an empty set. The simulation time is zero as well, as it has just begun, and the probability is 1.0 as that is the certain initial state of the model. Theoretically, depending on the distribution functions and size of the time step Δt, one of the following can happen:

1) Task1 completes,
2) Task3 completes, and
3) None of the tasks complete

According to this, following proxels will be generated:

1) $((Task4, Task3)(0, \Delta t), \{Task1\}, \Delta t, p11)$, $((Task6, Task3)(0, \Delta t), \{Task1\}, \Delta t, p12)$, $((C, Task3)(0, \Delta t), \{Task1\}, \Delta t, p13)$,
2) $((Task1, Task5)(\Delta t, 0), \{Task3\}, \Delta t, p21)$, $\big((Task1, C)(\Delta t, 0), \{Task3\}, \Delta t, p22\big)$, and
3) $((Task1, Task3), (\Delta t, \Delta t), \emptyset, \Delta t, 1 - p11 - p12 - p13 - p21 - p22)$

In case (1) and (2) there are more sub-cases due to the fuzzy rules, i.e. depending on the fuzzy membership function "*close to deadline*" value. Let us consider the proxel (1). The first case represents the possibility that Team A starts implementing Task4. The second case represents the possibility that Team A starts implementing Task6. Finally the third case represents the possibility that Team A is release because the project is close to deadline, as specified in the fuzzy rule. In that sense:

$$p13 = 1.0 \times \lambda 1(0) \Delta t \times \mu(0, \Delta - \tfrac{\Delta}{4}, \Delta),$$

where $\lambda 1(0)$ is the instantaneous rate function for the completion of Task1, and $\mu(0, \Delta - \frac{\Delta}{4}, \Delta)$ is the fuzzy membership function of "*close to deadline*".

This clearly illustrates the development of the proxel-based simulation. To obtain the final statistics we sum the probabilities for each discrete state of the model at every time step. The discrete state is composed of the tasks that each team is working on, along with the set of completed tasks.

C. Simulation Results

In the following we present the simulation results of our model. The proxel-based simulation provides complete results, i.e. a probability function of the duration of the project (see Figure 6) along with any needed statistics as is the case with the task completion probabilities. In addition, it provides the probability functions of having each of the tasks completed, as shown in Figure 7. The simulation shows that:

1) the three cancelable tasks, representing the three useful features we have specified, as expected have probabilities less than 1.0 of getting completed within the timeframe of the project. This is due to the fact that they can be canceled if the project is nearing the deadline, more realistic result.
2) Task6 has the lowest probability to be implemented within the given timeframe, whereas Task5 has the highest probability to be implemented within the given timeframe. This is slightly counter-intuitive, as one would expect that the task that can be accomplished by more teams has the highest probability of completion. This implies that the simulation can provide a significant insight into the real assumptions and behavior of the project schedule, thus impacting the feature selection, by providing additional information.

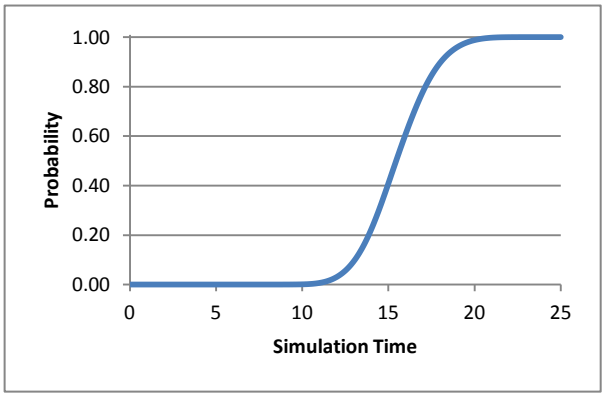

Figure 6: Probability of completing the project

3) Task5 can be implemented before the implementation of Task2, in spite of Task5 being a cancelable task. This is due to the precedence constraints we have. Recall that Task2 should be implemented after Task1. Hence, in order to optimize the usage of resources, once Team B finishes the implementation of Task3, it starts working on Task5 rather than staying idle waiting for Task1 to complete.

D. Discussion

As afore-described, our focus is to obtain the ranking of probabilities to have cancelable tasks completed within the project deadline. This provides us with an insight to aid the process of selection of useful features all along with the optimization of the human resources usage.

The enhanced project schedule model allows for including higher degree of uncertainty, while increasing the authenticity of the model and aiding the decision making. We have illustrated our approach with a simple model to aid the comprehension of the approach. However, the fuzzy rules can be much more complex, and theoretically, the proxel-based method can handle them easily. This is a work in progress, where the goal is to produce a tool to fully automate the process. In addition, simulation results depend on the initial state of the project schedule model. Hence, in order to have an accurate result, we need to consider all possible initial assignments of teams over the tasks.

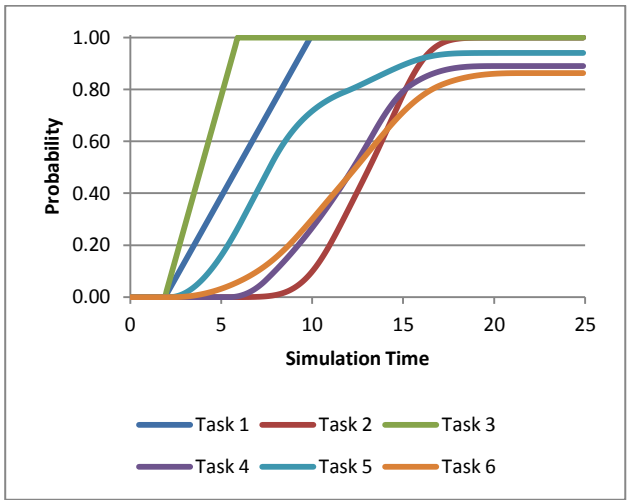

Figure 7: Probability of having each of the tasks completed

In the current model, our approach does not allow interruption of tasks, which we believe will make our model even more realistic. Certainly, not all tasks can be interrupted, it will be a feature of the task itself. The interruption of tasks will allow the modeling of requirements volatility, a major problem in software project.

Another considered future improvement is the team takeover, which occurs regularly and can be also managed by the fuzzy.

V. SUMMARY AND OUTLOOK

We presented a new simulation model that supports the process of useful features selection. The goal is to help the manager in selecting and prioritizing these nice-to-have features and to guide such selection based not solely on human judgment, but also on a robust simulation approach that takes into account additional uncertainty factors.

We consider features' priorities to be the input to the simulation tool. The simulation approach distinguishes between *useful* features that under some circumstances may be canceled, and the rest of them, for which the manager has no choice but to deliver them at the end of the project. Finally, we calculate probabilities of having each project task completed, which provides the manager with insight of realistic chances of having a feature implemented if the project plan deviates because of human resources uncertainties. The approach that we are presenting is not meant to be a comprehensive solution to features selection. Rather, it aims to augment the chance of a better selection of features based on feature priority and team availability.

Our model has three main advantages: 1) it targets the optimization of resources usage, and hence, minimizes idle time of teams; 2) it identifies the nice-to-have features that have the highest probability to be implemented within project/release deadlines and with respect to human resources uncertainties; and 3) it gives a new criterion for useful features selection. As a result, the obtained project baseline is expected to be of higher quality and depth.

In our future work, we aim at applying our approach in an industrial project planning. We also plan to extend the simulation model to represent more priority levels. And finally, we aim at investigating the extension of the model to handle multi-project resource sharing.

REFERENCES

[1] J. K. Pinto, *Project Management*, 2 ed., 2002.
[2] R. N. Charette. (2005). *Why Software Fails?* . Available: http://spectrum.ieee.org/computing/software/why-software-fails/
[3] N. F. Matta and R. N. Ashkenas, "Why good projects fail anyway," *Harvard Business Review*, vol. 81, pp. 109-116, 2003.
[4] P. J. Denning and R. D. Riehle, "The profession of IT Is software engineering engineering?," *Communications of the ACM*, vol. 52, pp. 24-26, 2009.
[5] J. F. Wateridge, "Delivering successful IS/IT projects: eight key elements from success

criteria to review via appropriate management, methodologies and teams," Brunel University, London, 2010.

[6] R. Leus, "The generation of stable project plans," *4OR: A Quarterly Journal of Operations Research*, vol. 2, pp. 251-254, 2004.

[7] D. Leffingwell and D. Widrig, *Managing Software Requirements : A Use Case Approach (2nd Edition)*: Addison wesley, 2003.

[8] K. E. wiergers. (1999) First Things First: Prioritizing Requirement. *Software Development*

[9] D. Port, *et al.*, "Using Simulation to Investigate Requirements Prioritization Strategies," presented at the Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering, 2008.

[10] S. Lazarova-Molnar, "The Proxel-Based Method: Formalisation, Analysis and Applications," Ph.D. Ph.D., Faculty of Informatics, University of Magdeburg, Magdeburg, 2005.

[11] C. Isensee and G. Horton, "Proxel-Based Simulation of Project Schedules," 2004.

[12] W. Huang, *et al.*, "Project Scheduling Problem for Software Development with Random Fuzzy Activity Duration Times," in *Advances in Neural Networks – ISNN 2009*. vol. 5552, W. Yu, *et al.*, Eds., ed: Springer Berlin / Heidelberg, 2009, pp. 60-69.

[13] M. J. Sobel, *et al.*, "Scheduling projects with stochastic activity duration to maximize expected net present value," *European Journal of Operational Research*, vol. 198, pp. 697-705, 2009.

[14] J. Navascu, *et al.*, "A Hybrid Model for Dynamic Simulation of Custom Software Projects in a Multiproject Environment," presented at the Proceedings of the International Conference on Software Process: Trustworthy Software Development Processes, Vancouver, B. C., Canada, 2009.

[15] S. Lazarova-Molnar and R. Mizouni, "A Framework for Enhanced Project Schedule Design to Aid Project Manager's Decision Making Processes," presented at the The 23rd European Modeling & Simulation Symposium, Rome, Italy, 2011.

[16] D. K. M. Mak and P. B. Kruchten, "NextMove: A Framework for Distributed Task Coordination," in *Software Engineering Conference, 2007. ASWEC 2007. 18th Australian*, 2007, pp. 399-408.

[17] A. Al-Emran, *et al.*, "Application of re-estimation in re-planning of software product releases," presented at the Proceedings of the 2010 international conference on New modeling concepts for today's software processes: software process, 2010.

[18] N. F. N. G. Amandeep, *et al.*, "Intelligent Support for Software Release Planning," in *Product Focused Software Process Improvement*. vol. 3009, F. Bomarius and H. Iida, Eds., ed: Springer Berlin / Heidelberg, 2004, pp. 248-262.

[19] S. Lazarova-Molnar and R. Mizouni, "Modeling Human Decision Behaviors for Accurate Prediction of Project Schedule Duration," in *Enterprise and Organizational Modeling and Simulation*. vol. 63, J. Barjis, Ed., ed: Springer Berlin Heidelberg, 2010, pp. 179-195.

[20] S. Lazarova-Molnar and R. Mizouni, "Floating Task: Introducing and Simulating a Higher Degree of Uncertainty in Project Schedules," presented at the IEEE Workshop on Collaborative Modeling and Simulation (CoMetS 2010), Greece, 2010.

[21] D. Firesmith, "Prioritizing Requirements," *Journal of Techology*, vol. 3, pp. 35-48, 2004.

[22] D. Dubois and H. Prade, "What are fuzzy rules and how to use them," *Fuzzy sets and systems*, vol. 84, pp. 169-185, 1996.

[23] T. P. Hong and J. B. Chen, "Processing individual fuzzy attributes for fuzzy rule induction," *Fuzzy sets and systems*, vol. 112, pp. 127-140, 2000.

[24] D. R. Cox, "The analysis of non-Markovian stochastic processes by the inclusion of supplementary variables," *Proceedings of the Cambridge Philosophical Society*, vol. 51, pp. 433-441, 1955.

[25] G. Horton, "A new paradigm for the numerical simulation of stochastic Petri nets with general firing times," *Proceedings of the European Simulation Symposium*, 2002.

[26] W. J. Stewart, *Introduction to the Numerical Solution of Markov Chains.*: Princeton University Press, 1994.

[27] F. J. Lin, *et al.*, "Protocol verification using reachability analysis: the state space explosion problem and relief strategies," *ACM SIGCOMM Computer Communication Review*, vol. 17, pp. 126-135, 1987.

[28] C. Isensee and G. Horton, "Approximation of Discrete Phase-Type Distributions," *Proceedings of the 38th annual Symposium on Simulation*, pp. 99-106, 2005.