Toward a Tool for Modeling and Managing Uncertainty in Project Planning

David Joslin
Computer Science and
Software Engineering Department
Seattle University
901 12th Avenue, P.O. Box 222000
Seattle, WA 98122-1090
joslind@seattleu.edu

Roshanak Roshandel
Computer Science and
Software Engineering Department
Seattle University
901 12th Avenue, P.O. Box 222000
Seattle, WA 98122-1090
roshanak@seattleu.edu

ABSTRACT

Good project managers recognize the inherent uncertainties that make project planning difficult. However, project planning tools often provide little support for managing uncertainties. We have developed a prototype tool for project plan analysis, using simulation and simple models for management decisions about resource reallocation. We have also developed some real-world data to use in evaluating our approach. Ultimately our goal is to develop a decision tool that would provide managers with better insights into the criticality of project tasks, as discovered by simulating the way the various uncertainties might unfold and interact as the project progresses. We recognize that some uncertainties are more benign than others, and we hope to provide managers with a tool that would help them to focus on the uncertainties that matter most.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management—*Time estimation*; I.6.5 [**Simulation and Modeling**]: Model Development

General Terms

Algorithms, Management

Keywords

Project management, project planning, uncertainty

1. INTRODUCTION

Critical path analysis can be useful in project management for estimating risk levels for tasks within a project. The critical path is defined as the set of tasks which collectively determine the total calendar time required for the project. Non-critical tasks have some amount of slack time, i.e., the degree to which variation in task duration does not impact the overall duration of the project. Given this, conventional wisdom says that critical tasks should be treated as more important than others when assigning resources.

Traditional critical path analysis techniques, such as the Critical Path Method (CPM) [5] offer simple deterministic methods that assume a fixed-time estimate for each task. Approaches such as Program Evaluation and Review Technique (PERT) [10] improve upon the simplest form of critical path analysis by taking into account that task durations cannot be predicted with certainty. However, this analysis still assumes static resource assignments, an assumption that is also not realistic. Managers will often reassign resources in response to new information about the expected duration of tasks, and may also decide to cancel lower-priority tasks entirely, such as when a proposed feature can be postponed to a later product release.

We believe therefore that a more realistic analysis of the criticality of tasks in a project needs to take both the uncertainty of task durations and the dynamic nature of resource reassignment into account. A task may be less problematic (or more problematic) than it might first appear to be because of the availability (or lack) of options for addressing delays by reassigning resources.

We envision a decision support tool that uses simulation techniques to help a project manager identify critical tasks and possible resource assignment strategies under a wide range of scenarios. This is only possible when taking the type of resource assignment adjustments a good manager might make into account. The key challenge in developing such a tool is that it must simulate, in a reasonably accurate fashion, the sorts of resource reassignment decisions an experienced project manager might make.

Simulating the intelligence, intuition, and domain expertise of a good manager is obviously not a reasonable goal to set. On the other hand, static resource allocation in the face of task duration estimates that change significantly is extremely unrealistic. Many software project managers use relatively simple heuristics for resources assignment in real world: will the assignment of a certain resource to a task decrease the duration and risk of that task, and if so by how much? We focus on studying models of resource reassignment heuristics that are simple but sufficiently realistic to provide useful results in a decision support tool.

IWLU 2007 Atlanta, Georgia USA

2. BACKGROUND AND RELATED WORK

There are two industry standard algorithms for determining criticality: the Critical Path Method (CPM) and the Program Evaluation and Review Technique (PERT). CPM was developed in the 1950's by the DuPont and Remington Rand corporations as a pencil-and-paper algorithm for determining the path in a project schedule that determines the minimum time possible to complete all tasks within the schedule. PERT was developed in 1958 by Booz-Allen-Hamilton, Inc. for the US Navy. It is different from CPM in using a weighted average of optimistic, expected, and pessimistic estimates to determine overall project duration.

The concept of probabilistic criticality is not a new one; [13], [2], [8] and [14] all address the concept. In [14], for example, the probability of a task being on the critical path is defined to be the task's criticality index. However, none of these efforts take the possibility of resource reassignment mid-task into account.

Simulation techniques have also been applied to project management on several occasions. Hebert [2] used Monte Carlo simulation techniques to address the Boolean problem in task criticality. However, Hebert assumes that once resource assignments have been made for a task those resources do not change until the task is complete, an unrealistic assumption in the software development industry.

Stutzke's work in software project estimation [12] is applicable in project simulation. Stutzke worked to refine Brooks' Law into a quantitative model that defines how much useful work additional staff can contribute to a task. In doing so he also defines the maximum number of resources that can be added before causing harm to the task, and how late in the task's life cycle these resources can be added. As we incorporate these quantitative models into our simulation, we expect to derive more realistic models in our managerial heuristic.

Joslin and Poole [4] applied Genetic Algorithm techniques originally developed as part of autonomous planetary rover planning research with NASA [3] to a simple project planning domain. Although the results were promising the focus of that earlier work was on generating project plans automatically, a focus that came out of the connection that work had to the autonomous rover planning problem. Simulation was used in the evaluation of the fitness function for the Genetic Algorithm, but the model was simplistic in its assumptions about heuristics for resource reallocation and assumptions about the effects of changing resource allocations.

To some extent our current work is a continuation of this earlier project, but with a very different emphasis. We continue with some of the basic ideas about simulation, but not the idea of applying genetic algorithms or other Artificial Intelligence techniques to the problem of project planning itself. We now instead focus on understanding how to do more realistic simulation with a goal of creating a tool that will give managers better insights into the most critical uncertainties they face.

3. OVERVIEW OF OUR APPROACH

For many projects, whether or not a task is critical depends on what has happened up to that point. However, the uncertainties faced during project planning make it difficult to estimate in advance how likely it is that a task will become critical. Simulation offers the possibility of providing such an estimate. Uncertainty in task duration is one key element in such a simulation, but using simulation also opens up the possibility of modeling other sources of uncertainty. For example, key employees often play multiple roles within an organization, and we may want to model the uncertainty about the percentage of time they will be able to devote to a project.

Whether or not a task is critical also depends on how a manager responds to events that affect it, and as mentioned above, we believe that any analysis that ignores the dynamic nature of project management will fail to accurately assess the criticality of tasks. If a task that appeared not to be critical in the original project plan suddenly is seen to be critical because the estimated duration for some other task increases, and if reassigning a resource to the newly-critical task will increase the chances of completing the project on time, a good manager will of course make that change.

A good manager also takes such contingencies into account to some extent in the creation of a project plan, even if the project planning software being used (if any) assumes static resource assignments. Some contingencies may be considered explicitly, by looking at a schedule and anticipating possible responses to the most likely scenarios in which the completion of the project might be delayed. Other contingencies may be considered more intuitively than explicitly, based on experience with the type of project and familiarity with the team members.

We obviously are not proposing to simulate the reasoning and intuitions of a good manager. We expect, however, that significant value from simulating dynamic resource allocation, compared to analysis that assumes static allocation, will be found in just being able to model the sort of cases in which resource reallocation is most clearly justified.

In designing such a model several questions come to mind. First, when would a reallocation of resources occur in the simulation timeline? We have a probabilistic model for the task duration, and an actual duration within each simulation run, but the simulated manager can only know the current estimate at any point in the simulation, not what the actual task duration will turn out to be later in that simulation. We therefore also model the change of duration estimates as time progresses within the simulation. These changes, like real-world status updates a manager would receive, are what may trigger a decision to reallocate resources.

We assume that changes to the estimated task duration become more likely as we get closer to the actual task duration within the simulation. Suppose the expected duration of some task, with current staffing, is eight weeks, with optimistic and pessimistic estimates of six and ten weeks respectively. In most cases, early completion of the task would not mean that after six weeks we suddenly discover that the task is complete, two weeks earlier than expected. It's much more likely that the estimates for the task would have been revised downward prior to that point. Similarly, if the task ends up requiring longer than expected it's unlikely that there would be no warning until the very day that eight weeks have elapsed. Our model attempts to capture this relationship between the actual duration the task will turn out to have and the points, if any, at which task duration estimates will be revised.

A second key question in this design is how the simulated manager decides whether resource reallocation will actually help. It would be very unrealistic to simply divide the to-

tal amount of work by the number of people assigned. Our model takes into account that not only is there a learning curve when someone is added to a task already in progress, their co-workers probably also become temporarily less effective as they help the new person come up to speed. Even after the learning curve effect has been absorbed into the schedule, the task duration does not decrease linearly with the number of people assigned, as Brooks famously observed [1], and we take this into account as well.

Within this outline a number of approaches to defining management heuristics are possible. Successful results in this effort to make the simulation more realistic may allow some later research into the use of Artificial Intelligence techniques for suggesting management strategies. Until then our goal is to look for effective resource reassignment heuristics that could be used in a simulation in order to provide a manager with better feedback as to the relative criticality of project tasks.

4. SIMULATION DESIGN

We have developed a new simulation framework that builds on the experience gained in our initial implementation [4]. This section provides a little more detail on that implementation.

Task representation. Tasks have the following attributes that are taken into consideration for simulation:

- Expected effort, the expected number of person hours required to complete the task.
- Pessimistic effort, the pessimistic expectation of effort.
- Presumed criticality, whether or not Microsoft Project considers this task critical.
- Required skill sets, a list of skills required to complete the task.

Task duration estimates. The simulation determines how long a task is actually going to take based on the estimates turned in to the project manager at the beginning of the project. Each estimate is made up of an optimistic, expected, and pessimistic estimate. The simulation determines the actual effort by selecting a value between the optimistic and pessimistic estimates, weighted by the implied confidence of the estimate. The confidence is inferred from the ratio of the pessimistic to expected estimates. As the confidence of the estimates go up, so does the probability of the actual duration of the task being closer to the expected estimate.

Task reassignment parameters. The learning curve is represented as a penalty associated with resource reassignment. When a resource is reassigned to a task, it is expected that there will be some period during which the resource is not operating at full capacity. For now, we use a simple cumulative Gaussian distribution.

In recognition of Brooks' Law [1] there is a penalty for assigning too many resources to a single task, modeled as an inverse cumulative Gaussian distribution. As more resources are added to a project, the less productive members of that team are, due to coordination and managerial overhead.

Task estimation uncertainty. The time granularity of the simulation is assumed to be daily, although any unit of time may be used. For each task, we have an initial estimate of the work required for completion, as discussed.

We do not assume that the original expected effort estimates are necessarily accurate, as very few software developers are perfectly accurate with their estimates. Consequently, during the simulation, the resource agents also report revised estimates. We are looking to model the real-world process of continuously revising estimates until the work is completed, as shown in McConnell's "Cone of Uncertainty" [6].

Developers tend to report they have slipped on a task as the deadline approaches; in other words, the closer one gets to a deadline, the more obvious it is that the deadline will not be met. To simulate this, we randomly select a point prior to the original expected estimate inversely proportional to the square of the distance from the expected estimate. When we reach this point in the simulation, the resource working on that task reports revised estimates based on the remaining actual number of days of work. This process continues until the error in the estimates decreases to less than a day.

Test data. We have created a data set to use in evaluating our simulation results. This data set was derived from a 14-month software development project at computer telephony company. This schedule is suitable for both performing simulation and comparison to the results of industry-standard CPM tools.

In order to determine the usefulness of the simulator and its results, the managerial heuristic must be evaluated. The simplest way to determine the quality of the heuristic is to see where the overall project duration of each run falls relative to the predicted duration of the original schedule.

We have made our sample project data available on-line [11] using files in the Attribute-Relation File Format (ARFF) [9]. There are four components to the data:

- Tasks: Descriptions, durations, project phases, presumed criticality, and required skills of all the tasks
- Resources: Names, seniority, and skills of all the available resources
- Task-Predecessor Map: Task interdependencies
- Task-Resource Map: Which resources were assigned to which tasks in the original schedule

See the ARFF files for specific information on the attributes expressed in the relations.

5. PROJECT STATUS

We have a functioning simulator written in Ruby, and one data set representing a medium-scale real-world project that we plan to use for initial testing. We have not yet done any significant testing with that data set, but we believe we are close to the point of being able to do so.

Significant features of the simulator include:

- Can import Microsoft Project files into the simulator.
- Uses Stutzke's work on quantifying Brooks' Law [12] to simulate the penalties associated with adding new resources to a task.
- Can optionally generate a Gantt chart that shows Van Slyke Criticality [14] as shades of red.
- Uses a plug-in model to facilitate experimentation with different managerial heuristics.

- Uses the common ARFF file format to encode project information, including tasks, resources, task predecessor maps, and resource assignments.

Other than heuristics implemented only for testing the framework, we have only implemented one very simple heuristic that attempts to capture some idea of how a manager might respond to changes in task duration estimates. In this heuristic, if a task's estimated completion time changes for the worse we check to see if the task is now critical. If so, we check to see if taking a resource from a non-critical, high-slack task would help, as evaluated against Stutzke's equations, and make that change only if it would help. We describe this only as a simple starting point, not as a proposal for a useful heuristic.

6. CONCLUSIONS AND FUTURE WORK

Better insights into the criticality of tasks can help software development managers cope with the uncertainties that they face in project planning. We believe that simulation, with sufficiently accurate models, can do a much better job of estimating task criticality than static analysis. We are currently focusing on the idea of simulating managerial heuristics for resource reallocation because the assumption of static task assignments is very unrealistic. We have also been attempting to refine our penalty models to conform to typical industry software projects; Stutzke's work [12] looks promising toward this goal.

As a starting point for evaluating resource reallocation heuristics we have developed some data for testing, based on real-world software development project, that we hope will help focus future work. We hope to find other real-world examples that could be used as well.

Eventually, we hope to develop a decision tool that would interact with an industry-standard project management tool such as Microsoft Project [7], in order to provide a criticality estimate for each task in a project. We also hope that the simulation data might be used to allow the manager to gain some insight into the sorts of scenarios that might cause tasks to become critical. For example, in addition to identifying a task that has some significant probability of becoming critical, we might be able to look back at the simulation runs in which it did become critical to determine if there are any common elements that would give the manager some insights into risks that could perhaps be mitigated before problems arise.

We don't envision such a tool being a substitute for the skills of a good development manager. However, managers must do project planning in the face of huge uncertainties about how the project will unfold, and better insights into the relative criticality of those many uncertainties can help a manager construct a more robust plan.

7. ACKNOWLEDGMENTS

The authors gratefully acknowledge the help of Stephen Silber in implementing the most recent version of the simulation code, in discussions of this work during the 2006-2007 school year, in collecting data from real-world projects and generating the test data described above [11], and in helping to write an early draft of this paper.

8. REFERENCES

[1] F. P. Brooks. *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley, 1982.

[2] J. E. Hebert. Applications of simulation in project management. In M. Spiegel, R. Shannon, and H. Highland, editors, *Proc. of the 1979 Winter Simulation Conference*, pages 211–219, 1979.

[3] D. Joslin, J. Frank, A. Jónsson, and D. Smith. Simulation-based planning for planetary rover experiments. In M. Kuhl, N. Steiger, F. Armstrong, and J. Joines, editors, *Proc. of the 2005 Winter Simulation Conference*, 2005.

[4] D. Joslin and W. Poole. Agent-based simulation for software project planning. In *WSC '05: Proceedings of the 37th conference on Winter simulation*, pages 1059–1066. Winter Simulation Conference, 2005.

[5] J. E. K. Jr. Critical path planning and scheduling: Mathematical basis. *Operations Research*, 9(3):296–320, 1961.

[6] S. McConnell. *Rapid Development: Taming Wild Software Schedules*. Microsoft Press, Redmond, WA, USA, 1996.

[7] Microsoft Corporation. Microsoft project, 2007. `http://office.microsoft.com/en-us/project/default.aspx`.

[8] M. T. Pich, C. H. Loch, and A. De Meyer. On uncertainty, ambiguity, and complexity in project management. *Management Science*, 48(8):1008–1023, 2002.

[9] P. Reutemann. Attribute-relation file format version 3.5.1, 2005. `http://weka.sourceforge.net/wekadoc/index.php/en:ARFF_%283.5.3%29`.

[10] D. D. Roman. The pert system: An appraisal of program evaluation and review technique. *The Journal of the Academy of Management*, 5(1):57–65, 1962.

[11] J. S. Silber. Sample project data, 2007. `http://www.seattleu.edu/scieng/comsci/ssrl/benchmark`.

[12] R. D. Stutzke. A mathematical expression of brooks' law. In *Ninth International Forum on COCOMO and Cost Modeling*, 1994.

[13] B. W. Taylor and L. J. Moore. R&D project planning with Q-GERT modeling and simulation. *Management Science*, 15(1):44–59, 1980.

[14] R. M. Van Slyke. Monte carlo methods and the pert problem. *Operations Research*, 11(5):839–860, 1963.