



Copyright Notice:

Materials published by Intaver Institute Inc. may not be published elsewhere without prior written consent of Intaver Institute Inc. Requests for permission to reproduce published materials should state where and how the material will be used.

Software Project Scheduling under Uncertainties

Intaver Institute Inc.
303, 6707, Elbow Drive S.W.
Calgary, AB, T2V0E5, Canada
tel: +1(403)692-2252
fax: +1(403)259-4533
sales@intaver.com
www.intaver.com

Managing of risk and uncertainties during the course of a project has become one of the priorities of the software project manager. Any research and development projects are affected by a large number of events, which can significantly change the course of a project. These events may form groups of related events or event chains. The paper discusses a proposed methodology of modeling the software project scheduling using event chains, classification of the events and chains, identification of critical chains, analysis of effect of the chains on project duration, cost, and chance of project completion. The paper presents a practical approach to modeling and visualizing event chains. The event chains methodology can contribute to reducing uncertainties in project scheduling through mitigation of philological biases and significant simplification of process of modeling, tracking, and analysis of project schedule.

Introduction

Project scheduling is an important step in the software development process. Software project managers often use scheduling to perform preliminary time and resource estimates, general guidance, and analysis of project alternatives. One of the major challenges in software project management is that it is difficult to adhere to the schedules due to the uncertainties related to requirements, schedules, personnel, tools, architectures, budgets, etc.

Software project managers recognize the importance of managing uncertainties. The iterative development process, identification and analysis of potential risks and utilization of other best practices can reduce uncertainties and help deliver the project according to the original time estimate, scope, and cost [1,7]. However, software project managers are not always familiar with probabilistic scheduling and tracking techniques or consider it as overhead. Modeling the project schedule with uncertainties on the planning phase remains important because it allows the manager to estimate feasibility of the delivery date, analyze the project, and plan risk mitigation.

This paper proposes a methodology for managing uncertainties based on an analysis of project events or groups of related events (event chains). The methodology can be

easily understood by project managers who are not familiar with advanced statistical theory. Managing uncertainties by the modeling of event chains is based on historical data, which leads to meaningful results. The software project scheduling using event chains methodology can be easily adapted. The implementation of the methodology does not require additional project management resources. In addition, off-the-shelf software tools that implement event chains methodology are available.

Overview of Existing Methodologies

Project planning usually starts with the development of work breakdown structure (WBS). The WBS is a hierarchical set of independent tasks. As a part of WBS, development costs and duration of tasks need to be estimated. After defining the set of tasks, project managers define the precedence relationship that exists among tasks. This information can be presented in the form of precedence networks and Gantt charts. The time needed to complete the project is defined by the longest path through the network. This path is called the critical path. Project managers can use the critical path method (CPM), which is available in most project management software, to identify the critical path [4,8].

In most cases, duration, start and finish time, cost, and other task parameters are uncertain. The PERT model (Program Evaluation and Review Technique) was developed in 1950s to address uncertainty in the estimation of project parameters. According to classic PERT, expected task duration is calculated as the weighted average of the most optimistic, the most pessimistic, and the most likely time estimates. The expected duration of any path on the precedence network can be found by summing up the expected durations. The main problem with classic PERT is that it gives accurate results only if there is a single dominant path through a precedence network. When a single path is not dominant, classic PERT usually provides overly optimistic results. Fortunately, most software projects have a single dominant path [3].

To overcome these challenges Monte Carlo simulations can be used as one of the alternatives. Monte Carlo simulations are a process that repeatedly sets values for each random variable by sampling from each variable's statistical distribution. The variables can be task duration, cost, start and finish time, etc. They are used to calculate the critical path, slack values, etc. Monte Carlo simulations have been proven an effective methodology for the analysis of project schedule with uncertainties. A number of software systems employ Monte Carlo simulations for projects [6]. However, Monte Carlo simulation is rarely used in software project management because of two main reasons. First, most software systems require at least some knowledge of statistics and risk analysis to define input data and interpret the results of the analysis. Second, Monte Carlo simulations for software development does not provide accurate estimates of project parameters (duration, finish time, cost, etc.) due to the greater uncertainties related to requirements, tools, resources, budget, etc. compared to many other industries.

Another approach to project scheduling with uncertainties was developed by Goldratt. Goldratt applied the theory of constraints (TOC) to project management [5,10]. The cornerstone of TOC is resource constrained critical path called a critical chain. Goldratt's approach is based on a deterministic critical path method. To deal with

uncertainties, Goldratt suggests using project buffers and encourages early task completion. Theory of constraints is well accepted in project management; however, the use of this approach in software development industry remains relatively low [4].

Software Project Management with Heuristics and Biases

The problem associated with all the aforementioned methodologies lies in the estimation of project input variables: task durations, start and finish times, cost, resources, etc. If input uncertainties are inaccurately estimated, it will lead to inaccurate results regardless of the methodology of project scheduling.

Tversky and Kahneman [14] have proposed that limitations in human mental processes cause people to employ various simplifying strategies to ease the burden of mentally processing information to make judgments and decisions. During the planning stage, software project managers rely on heuristics or rules of thumb to make estimations. Under many circumstances heuristics lead to predictably faulty judgments or cognitive biases.

Following are short descriptions of some heuristics that affect the estimation of project variables for software project management.

The availability heuristic [2,13] is a rule of thumb in which decision makers assess the probability of an event by the ease with which instances or occurrences can be brought to mind. For example, project managers sometimes estimate task duration based on similar tasks that have been previously completed. If they are making their judgment based on their most or least successful tasks, it can cause inaccurate estimation.

The anchoring heuristic [14] refers to the human tendency to remain close to the initial estimate. For example, anchoring will lead to an overestimation of the success rate of the project with multiple phases because the chance of completion of each separate phase of the project can be an anchor in estimating the success rate for the whole project [9].

Judgments concerning the probability of a scenario are influenced by amount and nature of details in the scenario in a way that is unrelated to the actual likelihood of the scenario [12]. It is called the representativeness heuristic. This heuristic can lead to the “gambler’s fallacy” or belief that a positive event is overdue because a series of negative or undesirable events have already occurred.

Decision makers can be exposed to many cognitive and motivational factors that can lead to biases in perceptions. This effect is often referred to as selective perception. For example, estimation of a task’s cost can be influenced by the intention to fit the task into the project’s budget. As a result, some of the project parameters can be overestimated.

Plous [11] has made some general recommendations for mitigating the negative impact of these and other heuristics. It is very important to keep accurate records and make estimations based on reliable historical data. Compound events should be broken into smaller events, which have known probabilities of occurrence. Discussion of best- or worst-case scenarios, for example the estimation of the most optimistic, the most likely, and the most pessimistic durations in PERT, can lead to unintended

anchoring effects. To reduce dependence on motivational factors, Plous recommends the analysis of problems without taking expectations into account.

Overview of Event Chains Methodology

The event chains methodology has been proposed to overcome difficulties associated with the estimation of project parameters, as well as to simplify the process of project scheduling with uncertainties for the software development.

According to the traditional project management methodology, the task (activity) is a continuous and uniform process. In reality, the task is affected by external events. These events can transform the task from one *state* to another. The state can be referred to as a process or part of the process with constant properties.

In most cases, especially for research and development projects such as software development, it is difficult to predict potential events at the stage of project planning and scheduling. Events can occur stochastically during the course of a task. One task can be affected many multiple probabilistic events defined by the event properties: chance of occurrence, probabilistic time of occurrence, and outcome (increase duration or cost, cancel task, assign or remove resource, etc.). These events will be included to the task's *list of events*. For example, during the course of development of the particular software feature, it may be discovered that the originally proposed software architecture is not appropriate. This discovery event may cause the cancellation of the feature or even the project. It can also cause an increase in the task duration and cost. The chance of occurrence of this event based on the previous experience of development of similar tasks is 20%. Based on the same historical data, the event should occur during first two weeks of the development.

In addition to *probabilistic events*, there are also *conditional events*. A conditional event will occur if some conditions, related to project variables, are met. For example, if the task has reached a deadline, the event "cancel task" can be generated. It is possible to have a combined conditional probabilistic event. For example, if the deadline is reached, there is 20% chance that the task will be canceled.

The events can significantly affect the tasks, a group of tasks, and the whole project. Tasks within a group can have different relationships. It can be a summary task with subtasks. A group may also include tasks with joint resources or other common parameters, which can be affected by the same events. It is important to identify groups of tasks in order to simplify the process of modeling with events.

One event can lead to other events or create event chains. For example, an event of architectural change in the software can require refactoring of the software component. As a result, the resource will be pulled from another task, which will change a state: a task will be delayed. Therefore, one event (architectural change) may cause a chain reaction and eventually lead to major change in schedule for the whole project. Event chains can be presented by an *event chains diagram*, as shown on Fig 1.

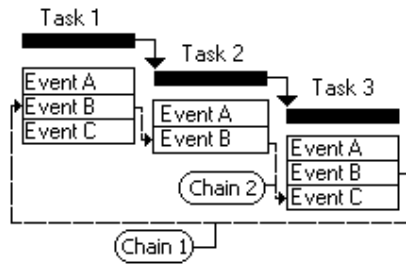


Fig. 1. Example of event chains diagram

Fundamentally, calculations in event chains methodology are a variation of Monte Carlo simulations. During the simulation process, project input variables (cost, duration, start and finish time, chance of completion) for each task will be calculated based on event properties. The result of calculation is a statistical distribution for the duration, start and finish time, success rate, and cost of the whole project or any separate task. The results can be represented in the form of frequency or cumulative probability plots. Statistical parameters for each output variable, including mean, variance, standard deviation, maximum and minimum values can also be calculated. They will be used to assess probability of completion of the project within a certain time and budget, as well as the probability of implementing a particular task (for example, features in a software development project).

All scheduling methods require making an initial estimate for the input project variables (task duration, start and finish time, etc.). Goldratt [5] recommends using median for the task duration; Monte Carlo simulations [6] allow the project manager to define a statistical distribution. Because event chains methodology is based on Monte Carlo simulations, a project manager is able to specify statistical distributions for the input project variables. However, it is not recommended because if event chains are defined, it can cause a double count of the same uncertainties. Instead, input parameters associated with *focused work on activity* or “best case scenario” should be defined. In addition, the project manager should define events and event chains that can affect the project schedule. For example, the manager can estimate that developing a particular feature will take from 5 to 8 days. Then the question that should be asked is, “What affects this duration?” It can be a number of potential events: requirement changes, unfamiliarity with development tools, unclear definitions of software architecture, hardware failure, etc. Lists of these events should be assigned to the task. If everything goes well and no issues occur (focused work on activity), the duration of the task will be 5 days.

The probability of a task lying on the critical path (criticality index) used in classic Monte Carlo simulation [8] also can be calculated as a part of the methodology. However, sometimes it is very important to find out which events or event chains affect output project variables the most. It can be accomplished using sensitivity analysis. These single events or event chains are called *critical events or event chains*. Results of sensitivity analysis can be presented in the form of sensitivity charts. To generate the sensitivity chart, correlation coefficients between output project parameters and events or event chains must be calculated.

One of the most important components of the event chains methodology is monitoring actual project performance and comparing it with original estimates. The simulation process must be repeated every time new results pertaining to the project or performance of each particular task have become available. Because events are time-

based, a new calculation will not include events that could have occurred prior to the actual time. As a result, a new updated project forecast would be available based on real project data.

Event chains methodology is designed to mitigate negative impact of heuristics related to estimation of project uncertainties:

1. The task duration, start and finish time, cost, and other project input parameters can be influenced by motivational factors such as total project duration to much greater extend than events and event chains. It happens because events cannot be easily translated into the duration, finish time, etc. Therefore, event chain methodology can help to mitigate certain effects of selective perception in project management.
2. The event chains methodology relies on estimation of duration based on focused work on activity and does not necessarily require low, base, and high estimation or statistical distribution; therefore, the negative effect of anchoring can be mitigated.
3. The probability of event can be easily calculated based on historical data. It helps to mitigate the effect of the availability heuristic. The probability equals the number of times an event actually occurred in previous projects divided by total number of situations when event could have occurred. In classic Monte Carlo simulations, the statistical distribution of input parameters can also be obtained from the historical data; however, the procedure is more complicated and rarely used in practical software project management.
4. The compound events can be easy broken into smaller events. Information about these small events can be supported by reliable historical data. This mitigates the effect of biases in estimation of probability and risk.

Single Events

Single events are the building blocks of the comprehensive probabilistic model of the software development process.

Each event has a number of properties. The events can affect the whole project, a group of tasks, a particular task, or the resource. For example, if it is discovered that a selected software tool does not provide the required functionalities, all tasks that are using this tool can be delayed.

The following types of events are commonly used in the software development project:

- Start and end tasks or group of tasks,
- Duration of a task or duration of each task within the group can be increased or reduced,
- Costs associated with a task or group of tasks can be increased (reduced),
- Tasks or each task within a group can be canceled,
- Resources can be reassigned or a new resource can be assigned, and
- Whole projects can be canceled.

A new task duration or cost can be calculated in different ways. The task can be restarted from the moment when an event has occurred. Further, the task can be delayed or duration can be increased/ reduced. For example, duration can be increased by 20%.

The events can be categorized based on relationship between individual tasks (group of tasks) they are assigned to and the tasks (group of task) they are affecting. The event can be assigned to and affect the same task. Alternatively, the event can affect a different task or a group of tasks from the task it was assigned to. For example, a purchase of more powerful hardware will reduce development time for a group tasks. Often a single event can be initiated within a project without any relationship to the particular task. It can affect a single task, a group of tasks, or a complete project. For instance, changes in the project's budget can affect all tasks from the moment these changes have occurred.

Another property of the event is the chance of its occurrence. For example, there is a 2% chance of the event where the whole project will be canceled due to budgetary constraints. If the cost or duration of the task has been increased or reduced, the event will include additional set of properties. This information includes time or cost additions or time and cost savings. This can be calculated in absolute units (days, dollars, etc.) or as a percentage of the task duration or cost. For example, in event of inconsistent software development requirements, duration of the construction iteration can increase by 30%.

One task can have a group of mutually exclusive events. For instance, there is a 20% chance that duration of a task will be increased by 35%, a 30% chance that duration will increase by 10%, and a 5% chance that task will have to be canceled. Alternatively, the task can be simultaneously affected by some combination of these events. For example, there is a 20% chance that duration and cost can be increased together.

The next property of the event is chronological. This parameter can be deterministic, but in most cases it is probabilistic. For example, the event can occur between the start time and end time of the task minus two days, but will most likely occur two weeks after the task has started. This information can be represented by the triangular statistical distribution.

The time when the event occurs is important. If the event results in the cancellation of the task, to calculate the task duration, it is important to know when it occurred. This information is also crucial when tracking of project performance in order to filter events that could have occurred before the actual date. Finally, in certain cases, it is essential to know when the event has occurred to calculate the new duration and cost.

Event Chains

Event chains are the cornerstones of the proposed methodology. There are two main groups of event chains: *explicit* and *implicit*. In explicit event chains, one event causes another event. Implicit event chains include conditional events; therefore, in implicit event chains, one probabilistic event may cause another event. For example, the original event affects task duration and if there is a change in requirements, task duration can be increased. However, the task may have a deadline. A conditional

event can be linked to this project parameter. If the deadline is reached, the event will result in cancellation of the task. In current example, the original event is causing the chain reaction, which leads to termination of the task.

The proposed methodology enables project managers to model very complex project scenarios. To illustrate, below provided some of the possible situations that can be modeled easier by using proposed methodology compared to traditional methods.

One of these scenarios relates to probabilistic and conditional branching, used in classical Monte Carlo simulations. For example, there is a 40% chance that a particular feature should be developed after analysis of the requirements, and a 60% chance that this development will not be necessary. Event chains methodology makes conditional and probabilistic branching much more flexible. For example, if one event has occurred, there will be a 60% chance that it will trigger an event in another task.

An event can activate other events immediately or with delay. Sometimes events can affect a future task. For instance, changes in requirements will lead to extra development in the future.

Sometimes, events can affect previous tasks in the project schedule. This is a common occurrence in software development, where an existing component requires refactoring to comply with a planned development. In this case, the refactoring task, which originally was not in the project schedule, will be automatically generated using the event “start task”. It leads to an important feature of the proposed methodology – the ability to reschedule activities using *dynamically generated tasks*.

Event chains can be modeled using *circular relationships*. Circular relationships are not just mathematical phenomena for the proposed methodology, they occur in the real world. For example, the development of a particular feature can fail because of a problem with software performance. To fix the performance problem, the developer must refactor an existing module. After this, the provability of failure of the performance test can be reduced. However, there is still a chance that the existing module must be refactored again. Project constraints such as deadlines and costs can be used to address the circular relationship problems.

In traditional methodologies, there are no relationships between tasks during the course of the task. In reality, *synergies* between tasks significantly affect project schedule. Event chains allow taking potential synergies into account. For example, if there is a delay in one task, other parallel tasks can be delayed. The scenario can be modeled by an event chain initiated by single event with an increased duration outcome.

In addition, event chains offer a possible solution to the resource allocation problem. This can be accomplished using the conditional events “Assign new resource” or “Reassign resource”, which are linked to the task deadline or an intermediate milestone. If there is a delay in a certain task caused by a specific event, new resources can be borrowed from another task and reassigned from the moment the event occurred. The task will then change its state and the project schedule will be recalculated with a new resource allocation. Simulation results will present the statistics of the resource allocations.

Analysis of Software Project using Event Chains Methodology

The planning, tracking, and analysis of software development projects is comprised of multiple steps. Generally, the process is similar to traditional approach; however, there are a few significant differences. The following example illustrates the workflow based on event chains methodology. For simplification, only single probabilistic events are included in this example. The model has been created using a commercial project planning software that utilizes event chains methodology.

Creating the Baseline Project Schedule

The first step in scheduling processes using event chains is very similar to what project managers do using traditional methodologies. The project schedule will be created and presented in the form of a Gantt chart. The project manager should specify input project parameters, such as duration, start and finish time, cost, etc., that are associated with a “best case scenario” or a focused work on activity.

Defining events

Each task can be affected by multiple potential events. The project manager should set up a list of events for tasks and resources. Fig. 2 shows list of events for the task.

	Risk name	Chance	Outcome	Start	Most Lik...	Finish	Chart
1	<input type="checkbox"/> Staff						
2	<input checked="" type="checkbox"/> Limited user interface expertis	15.0 %	Restart task	5.0 %	50.0 %	95.0 %	
3	<input checked="" type="checkbox"/> Limited expertise in client/ser	10.0 %	Increase duration	5.0 %	50.0 %	95.0 %	
4	<input type="checkbox"/> Technical						
5	<input checked="" type="checkbox"/> Wrong selection of developm	1.0 %	Cancel project	0.0 %		100.0 %	
6	<input checked="" type="checkbox"/> Software performance not act	0.1 %	Cancel project	0.0 %		100.0 %	
7	<input type="checkbox"/> Organization Management						
8	<input checked="" type="checkbox"/> Budgetary problems	0.1 %	Cancel project	0.0 %		100.0 %	
9	<input checked="" type="checkbox"/> Management changes	1	0.1 %	Cancel project	0.0 %	100.0 %	
10	<input checked="" type="checkbox"/> Management changes	2	0.2 %	Cancel task + summary tasks	0.0 %	100.0 %	
11	<input type="checkbox"/> Requirements						
12	<input checked="" type="checkbox"/> Early requirements changes	40.0 %	Restart task	0.0 %	35.0 %	50.0 %	
13	<input checked="" type="checkbox"/> Later requirements changes		start task	50.0 %	75.0 %	100.0 %	

Mutually exclusive events

Fig. 2. Hierarchical event table

Each event has a number of properties. The process of defining these events can be tedious and complicated. To simplify the definition of events, event templates can be used. An event template is a standard hierarchical list of events for a particular industry or the group of projects. Lists of events for the task, group of tasks, or a project can be generated based on a template. To simplify the management of events even further, events from the template can be turned on and off on a task-by-task basis.

Performing Simulation and Analysis

To generate a schedule with uncertainties, Monte Carlo simulations should be performed using a baseline project schedule and an event list. The number of simulations can be defined based on the lowest probability of the occurrence of events. The simulation can be stopped when the results of simulations converge: that is, when the main calculation outputs (duration, finish time, project cost, etc.) within a given number of simulations remain close to each other. Unfortunately, because of the discrete nature of the event chains, simulations will converge relatively slowly. In reality, the number of simulations can be between a few hundred to a few thousand. However, using modern computer hardware, Monte Carlo simulations for realistic software development projects can be executed within seconds. Actual simulation time depends on computer performance, number of simulations, number of tasks, and number of events.

The results of a calculation can be presented in the form of a Gantt chart together with baseline project schedule (see Fig. 3).

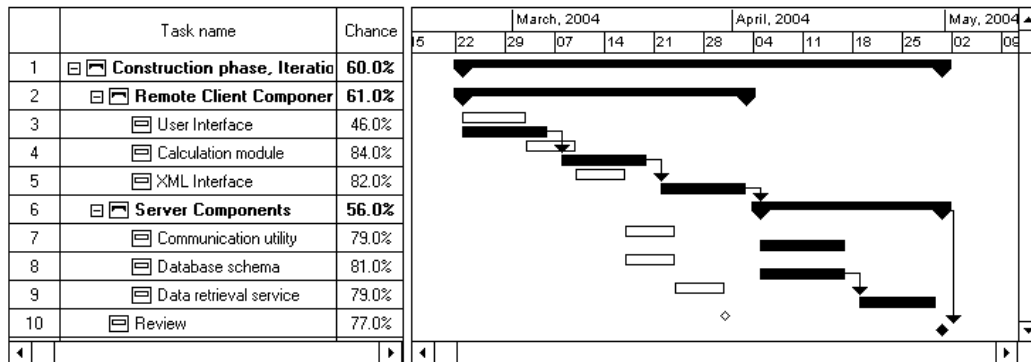


Fig. 3. Results of calculation and baseline project schedule

In this example, events significantly increased the duration of all tasks and the whole project. Results of the simulation are shown on Fig 4. as a table and a frequency chart. The chance that project duration is below a certain number is a measure of the project risk.

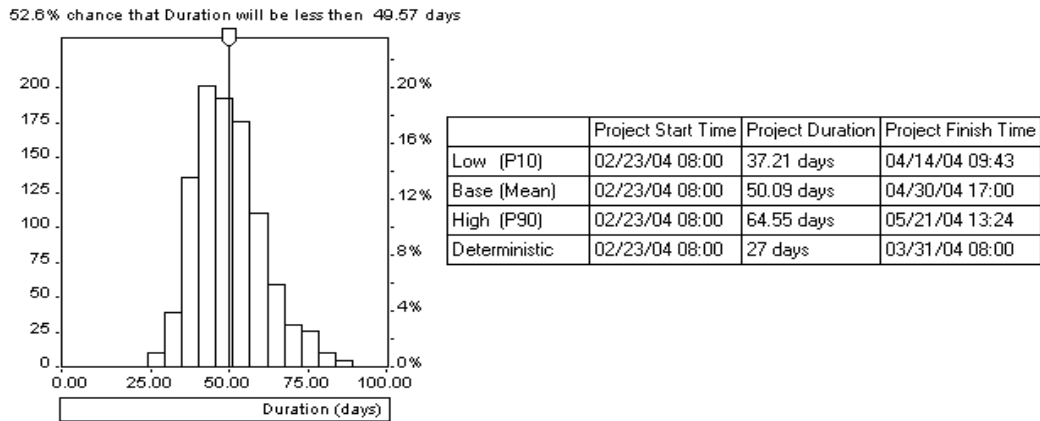


Fig. 4. Simulation results: frequency chart for duration and results in table format

Results of the sensitivity analysis are presented on Fig. 5. The chart shows how sensitive the project duration is to the uncertainty related to a number of events. It demonstrates that duration is most sensitive to the “Software Performance Not Acceptable” event. This means that software performance is the project’s the most important risk factor. This example illustrates how the proposed methodology allows the generation of risk lists for the software projects.

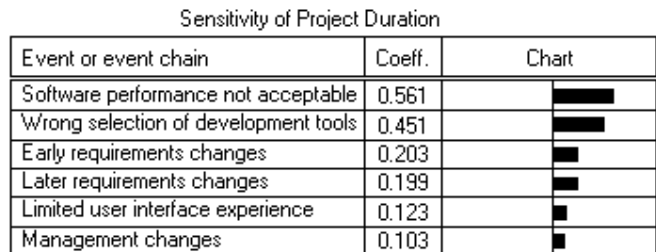


Fig. 5. Sensitivity chart for events and event chains

Monitoring the Course of the Project

To track project performance, the project manager should input the completion percentage of a particular task and the date and time when this measurement occurred. The results of tracking a specific task are shown on Fig. 6. Using this chart, the manager can easily compare actual data with the baseline and calculated results. As soon as the project manager inputs a new percentage of work done, a Monte Carlo simulation can be performed and a new forecasted task duration and finish time will be calculated.

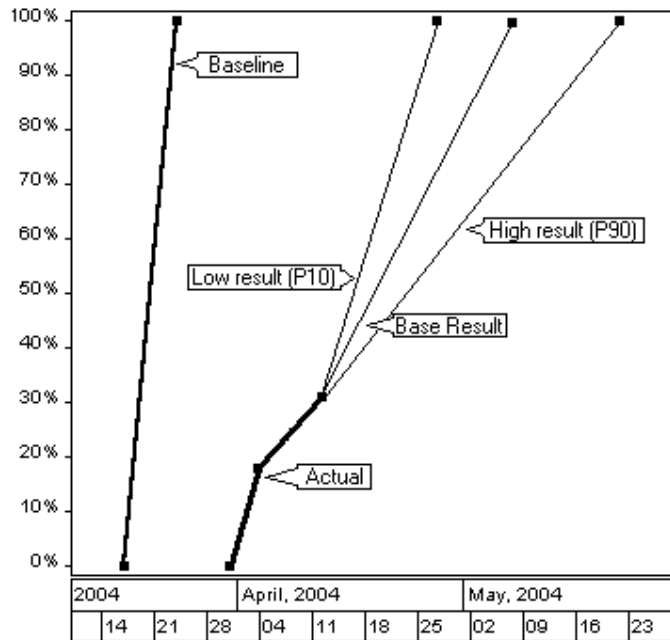


Fig. 6. Tracking and forecasting of performance for the specific task

Conclusions

The proposed event chains methodology is applicable to different time-related business or technological processes. The methodology can be very effective in software project management, where it can significantly simplify a process with multiple uncertainties.

Event chains methodology includes the following main principles:

1. A task in most real projects is not a continuous uniform process. It is affected by the external events, which transform task from one state to another.
2. The events can cause other events, which will create the event chains. These event chains will significantly affect the course of the project.
3. The identification of the critical chain of events makes it possible to mitigate their negative affects. Risk list of the project can be generated as a result of sensitivity analysis.
4. The tracking of the task progress and the continuous comparison of actual progress with estimates ensures the project schedule is calculated based on updated information.
5. The analysis of historical data related to current and past similar project performance is necessary to obtain information about probabilities and outcome of events.
6. The event chains methodology simplifies rescheduling, conditional and probabilistic branching, analysis of project with synergists and circular dependencies, and resource allocation.

Event chains methodology is a practical approach to scheduling software projects that contain multiple uncertainties. A process that utilizes this methodology can be easily used in different projects, regardless size and complexity, using off-the-shelf software tools.

References

1. Beck K.: *Extreme Programming Explained – Embrace Challenge*. Addison-Wesley, Reading, MA. (2000)
2. Carroll, J.S.: The effect of imagining an event on expectations for the event: An interpretation in terms of availability heuristic. *Journal of Experimental Psychology*, 17, (1978) 88-96
3. Cho J.G., Yum B.J.: An Uncertainty Importance Measure of Activities in PERT Networks. *International Journal of Production Research*, 12, (1964) 460-470
4. Futrell R.T., Shafer D.F., Shafer L.I.: *Quality Software Project Management*, Prentice Hall PTR, Upper Saddle River, NJ, (2002)
5. Goldratt, E.: *Critical Chain*. North River Press, Great Barrington, Mass. (1997)
6. Hulett D.T.: Schedule Risk Simplified, *PM Network*, July, (1996) 23-30
7. Jeffries, R., Anderson A., Hendrickson C.: *Extreme Programming Installed*. Addison-Wesley, Reading MA. (2000)
8. Klastorin T.: *Project Management. Tools and Trade-Offs*. Wiley, (2004)
9. McCray G.E., Purvis R.L., McCray C.G.: Project Management Under Uncertainties: The Impact of Heuristics and Biases. *Project Management Journal*. Vol. 33, No. 1. (2002) 49-57
10. Newbold R.C.: *Project Management in the Fast Lane: Applying the Theory of Constraints*. St. Lucie Press, Boca Raton, FL, (1998)
11. Plous S.: *The Psychology of Judgment and Decision Making*, McGraw-Hill (1993)
12. Tversky, A., Kahneman, D.: Belief in the law of small numbers. *Psychological Bulletin*, 76, (1971) 105-110
13. Tversky, A., Kahneman, D.: Availability: A heuristic for judging frequency and probability. *Cognitive Physiology*, 5, (1973) 207-232
14. Tversky, A., Kahneman, D.: Judgment Under Uncertainty: Heuristics and biases. *Science*, 185, (1974) 1124-1130