

# A Critical Examination of Evidence-Based Scheduling

Russell Thackston

PhD Student, Department of Computer Science and Software Engineering, Auburn University

Russell.Thackston@Auburn.edu

## Abstract

Scheduling and schedule estimates are one of the greatest challenges in software project management. Project managers are commonly asked to predict delivery dates at a very early stage of the software development lifecycle, many times before the necessary details are available to formulate an educated response. Fog Creek Software, a Manhattan-based shrink-wrap software firm has developed a method known as Evidence-Based Scheduling (EBS) for predicting software development schedules through the use of expert opinion and historical performance. Rather than approaching scheduling as a problem with a single solution, EBS predicts a range of completion dates and the probability of each date being the actual completion date. The main strengths of EBS lie in its simplicity and ease of use, once the EBS process is fully understood by its users. Like many approaches to project scheduling, EBS has its share of weaknesses, such as a heavy reliance on “expert opinion,” however, EBS could represent a significant step forward for organizations lacking a formal schedule estimation process.

**Keywords:** Evidence-Based Scheduling, Software Process, Scheduling, Software Engineering, Monte Carlo Simulation, Practice-Centered Software Engineering

## Overview

In the early 2000's, Fog Creek Software developed an estimation process known as Evidence-Based Scheduling (EBS) to address their difficulties in estimating software development schedules. EBS feeds historical data and expert opinion, on a developer-by-developer basis, into a Monte Carlo-style simulation to produce a probability curve representing the confidence of meeting a particular schedule. Fog Creek's co-founder, Joel Spolsky, claims that EBS is extremely effective as an in-house estimation tool and has made EBS as a cornerstone feature of their flagship product Fog-Bugz. This paper will examine the foundational concepts of EBS as they relate to the software development process and will explore which aspects of EBS that make it particularly suitable (or unsuitable) for use outside Fog Creek Software in a variety of environments. This paper will also specifically examine the potential use of EBS in the Practice-Centered Software Engineering (PCSE) process.

### *Fog Creek Software – About the Company*

Fog Creek Software was founded in September 2000 by Joel Spolsky and Michael Pryor<sup>1</sup>. The company's core products include:

- *FogBugz* – An issue tracking, project planning, and scheduling application written in a custom language, *Wasabi*, which compiles to both .NET and PHP platforms.
- *Copilot* – A web-based, consumer-oriented, remote desktop application hosted by Fog Creek Software and based on the open source TightVNC product with C#.NET web components.
- *CityDesk* – A client-side, proprietary content management system written for Windows platforms.

The company's core strengths clearly lay in the domain of Microsoft development tools, which is not surprising considering Joel Spolsky worked on the Excel team at Microsoft in the 1990's. Furthermore, the company focuses on developing, marketing, and supporting proprietary applications. As such, internal management and external (indirect) market forces drive their schedules and processes, rather than third-party entities, such as clients. As a result, Fog Creek Software is free to develop and use proprietary versions of standard software processes without being accountable to others.

## Evidence-Based Scheduling

Evidence based scheduling is a method of predicting software development schedules by building a probability curve representing possible completion dates derived from three major components:

- Historical data
- Initial schedule
- Monte Carlo-style simulations

The historical data is typically gathered from each developer's past time sheets and includes their time estimates and actual development time for prior work. An initial schedule for the new development work is prepared based on the conceptual design and the new set of estimates provided by each developer. A Monte Carlo style simulation is used to generate a set of possible completion dates with corresponding probabilities.

### *Historical Data*

EBS uses historical data of estimated and actual durations as a predictor of future performance. Most schedules begin with an estimate of how long it will take to complete each particular task. The total of all tasks, plus overhead, equals the predicted duration. Many processes rely on expert opinion or individual developer estimates to determine each task's estimated duration.

EBS relies on individual developers to analyze their assigned tasks and to create estimates for those tasks' expected durations. As the developer works on a particular task, their actual time is tracked, usually via a time sheet. This historical data provides some insight into the accuracy, or inaccuracy, of a particular developer's estimates. Table 1 - Sample Developer Estimated and Actual Hours shows a developer who consistently underestimates tasks by a few hours.

**Table 1 - Sample Developer Estimated and Actual Hours**

Task	Estimated	Actual
1	4	5.9
2	6	7.4
3	4	5.1
4	8	8.5
5	12	13.9

### Initial Schedule

As stated, an initial schedule for the new development work is prepared based on the conceptual design and the new set of estimates provided by each developer. At this point, the accuracy of the initial schedule is wholly dependent upon the accuracy of each developer's estimates. Many factors may come into play that can affect the actual development time. These factors may include, but are not limited to:

- Individual developer estimating skills
- Environmental influences (holidays, illnesses, etc.)
- Work environment (productivity, interruptions, etc.)
- Quality of the conceptual design

EBS does not attempt to impose greater accuracy on the developer's estimates, and thereby the schedule. Instead, EBS attempts to predict the probability of completing development on a particular date.

### Weighting Individual Estimates with Historical Data

Let's assume a particular developer has completed ten tasks and the developer's estimated durations and actual durations were recorded as shown in Table 2 - Sample Developers Velocities. For each of the estimated and actual durations the *velocity* is calculated as  $[Estimated\ Duration] \div [Actual\ Duration]$ . The closer a velocity is to one (1.0) the more accurate the developer's estimate. Velocities below one indicate the developer's estimate was low and, conversely, velocities above one indicate the developer's estimate was too high.

**Table 2 - Sample Developers Velocities**

Task	Estimated	Actual	Velocity
1	4	5.9	0.68
2	6	7.4	0.81
3	4	5.1	0.78
4	8	8.5	0.94
5	12	13.9	0.86
6	10	11.9	0.84
7	8	8.2	0.98
8	15	15.7	0.96
9	5	5.4	0.93
10	3	4.6	0.65

As you can see in this example, this developer consistently underestimates the required effort, as represented by velocities between 0.65 and 0.98. These velocities represent the developer's historical accuracy. If we assume that the developer's future performance is likely to match their historical performance, then we can take a future estimate (eight hours, for example) and divide it by the (sorted) range of velocities and get the values in the third column of Table 3 - Sample Adjusted Estimate). Assuming all outcomes are equally likely, there is a 10% chance of being done in 8.2 hours and an 80% chance of being done after 10.3 hours, since eight out of the ten possibilities occur by the time 10.3 hours have elapsed.

**Table 3 - Sample Adjusted Estimate (Sorted by Velocity)**

Task	Velocity	Adjusted (From 8 hrs.)	Probability (Cumulative)
7	0.98	8.2	10%
8	0.96	8.3	20%
4	0.94	8.5	30%
9	0.93	8.6	40%
5	0.86	9.3	50%
6	0.84	9.5	60%
2	0.81	9.9	70%
3	0.78	10.3	80%
1	0.68	11.8	90%
10	0.65	12.3	100%

### Creating a Weighted Schedule

At this point, for any particular task, we can see a series of possible outcomes and the probability of each occurring. This can then be translated to a set of possible outcomes by accumulating all the possible outcomes and their probabilities for the entire project. However, this presents an extremely large number of combinations of possible outcomes. For example, a project with 120 tasks, assigned to a team of developers with 10 historical velocities, gives  $10^{120}$  possible outcomes, each with their own probability of occurring.

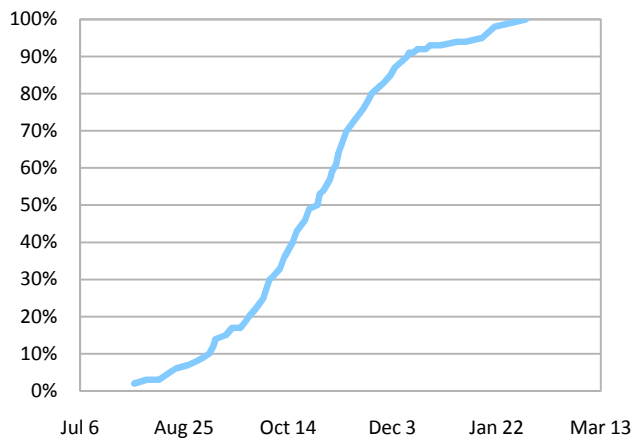
Instead of attempting to calculate all possible outcomes, a Monte Carlo style simulation is used to produce a relatively large, random set of possible outcomes. This subset of possible outcomes is representative of all possible outcomes. The following section discusses in greater detail how a Monte Carlo simulation is constructed and applied to EBS.

### Monte Carlo Simulations

In a Monte Carlo simulation, repeated random sampling is used to compute a result, or set of results, that are difficult or impossible to determine exactly through a deterministic algorithm.<sup>2</sup> In EBS, rather than giving an exact completion date, the Monte Carlo simulation produces a series of possible completion dates where the distribution of values is weighted based on historical data. The end result is a distribution of possible completion dates that demonstrate the **probability** of a particular date being met.

Specifically, a *single simulation* would (1) iterate through all tasks and for each task, (2) randomly select a velocity from the assigned developer's historical velocities to represent the accuracy of the estimate and, (3) produce an adjusted duration for the task by dividing the duration by the selected velocity. All durations for all tasks would then be summed to produce a single *possible* duration from the project.

This single simulation would be run over and over again until a relatively large set of possible project durations has been produced. In the case of EBS, about one thousand simulations are run. The results of the simulations are displayed on a graph (see Figure 1 - Sample Probability Curve). From this graph, the probability of meeting a particular date can be determined.



**Figure 1 - Sample Probability Curve**

### *Interpreting the Results*

The graph in Figure 1 - Sample Probability Curve demonstrates that there is about an 80% probability of the software being delivered on October 30<sup>th</sup>. It would be impossible to know the exact date all work on the software will be completed. However, by taking into account a developer's historical performance with regard to estimating, we can associate confidence with any particular date being met.

In business terms, if the project manager is asked for the likelihood of completing the project on October 10<sup>th</sup>, they may respond that "in 1,000 simulations, only 250 of the simulations completed on or before that date." Similarly, the project manager may say that 100% of the simulations showed a completion date prior to January 30<sup>th</sup>.

### **Aspects of EBS**

Before implementing EBS, an organization will need to satisfy certain prerequisites, including compiling historical data, preparing the conceptual design, and gathering developer estimates for tasks. This information allows EBS to produce an initial probability graph showing confidence in particular dates. EBS can also produce a completion date confidence graph showing confidence changes over time. Lastly, EBS can produce various graphs to help categorize and improve individual project team members' estimation skills.

### *Prerequisites to EBS*

Evidence Based Scheduling requires a series of inputs, including historical data and an initial schedule. A detailed examination of these inputs reveals some specific characteristics. Furthermore, these inputs may require certain characteristics relating to organizational structure and culture of the organization implementing EBS.

### **Historical Data**

Historical data is commonly garnered from developer's time sheets and includes estimated and actual elapsed time for each task performed.<sup>3</sup> This historical data should go back several months, at a minimum, and the granularity should be analogous to that of future projects.

Note that organizations with certain characteristics may find it difficult to implement EBS. One characteristic is employee turno-

ver. Fog Creek software goes to great lengths to hire and retain quality developers<sup>1</sup>. This ensures a level of consistency from project to project. Typical organizations, however, experience higher levels of employee turnover. This turnover can make it difficult or time-consuming to accumulate a significant set of velocity values. Furthermore, large organizations can experience intra-departmental turnover. Historical data must be tracked and transferred between departments when this occurs.

In the absence of significant historical data, an organization should err on the side of caution and assume poor estimating skills on the part of the developer. The net effect of such an assumption will be to reduce confidence in particular dates and push schedules toward a longer timeframe, which is more likely to occur than shorter schedules.

### **Conceptual Designs and Initial Schedules**

An initial schedule for the upcoming project is assembled from the conceptual design and each developer's estimates for their assigned tasks. Of course, this requires a conceptual design, detailed work breakdown structure, task assignments to developers, and developer estimates, all of which is no different from typical software project planning. The key point for EBS is that each task should be no larger than sixteen hours.<sup>4</sup> Tasks larger than sixteen hours are seen as too coarse grained and should be further decomposed.

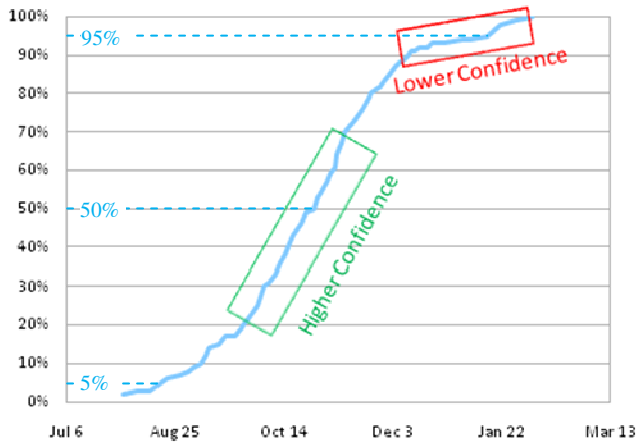
A characteristic of an organization which can affect the quality of an EBS implementation is team leadership. Many organizations utilize a pool of resources to draw from. This pool may include developers, analysts, and architects. To effectively use EBS, the architects must use a consistent approach to constructing a conceptual design and initial schedule. If the style, quality, and/or granularity of these deliverables varies widely, then individual developer velocities will also vary, based on these inputs, reducing the quality of the EBS outputs.

For example, a developer may be very good at estimating small tasks, but not as good at estimating larger ones. Each time the developer works with one architect, who consistently uses a fine-grained approach to task decomposition, the developer's estimates are very accurate. However, when the developer works with a second architect who leans toward coarse-grained task decomposition, the developer's estimates are consistently off.

Overall, EBS will account for this with greater uncertainty in fine-grained projects and with slight optimism in coarse-grained projects. However, this affects the probabilities associated with dates by making them less accurate, when the accuracy could be improved by consistent architectural designs.

### *Outputs of EBS*

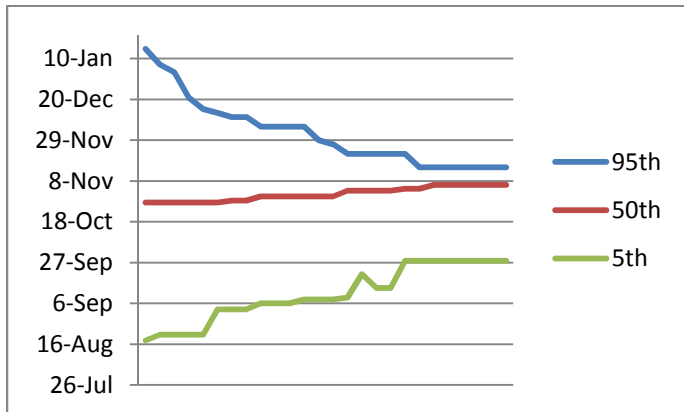
The primary output of EBS is a probability graph which displays the probability of completing the project on any particular date (see Figure 1). Portions of the graph with a high slope (more vertical line) represent higher confidence, whereas portions of the graph with a lower slope (more horizontal line) represent lower confidence. Figure 2 - Relative Confidence demonstrates how easy it is to pick out these areas. For example, a project manager may infer that the 20-70% probabilities are relatively accurate, whereas the 90-100% probabilities are not.



**Figure 2 - Relative Confidence**

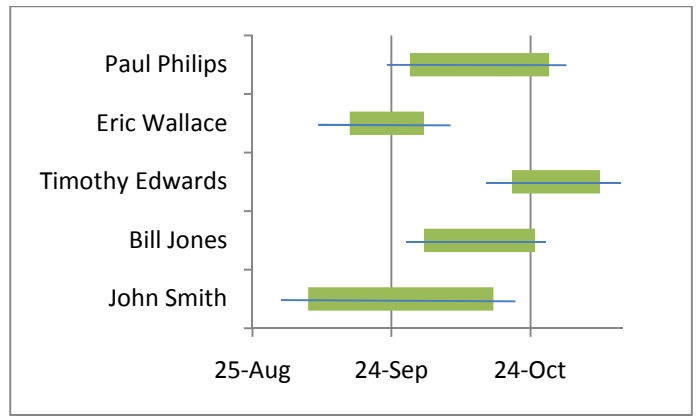
A secondary output of EBS is a graph showing completion date confidence over time (see Figure 3 - Completion Date Confidence Graph). This graph is constructed using the 95<sup>th</sup>, 50<sup>th</sup>, and 5<sup>th</sup> percentile prediction each day as work progresses (see Figure 2 annotations). Theoretically, these values should converge toward a single point as the completion date approaches, a concept reinforced by the uncertainty factor in estimating, developed by NASA<sup>5</sup> and referred to as the *cone of uncertainty*.

Figure 2 - Relative Confidence shows a project with an estimated completion date converging toward a range of late September to early October. As the project progresses and more of the tasks are completed, the graph will continue to converge toward the actual completion date. Although the project manager cannot be certain that the project will complete in this time frame (late September to early October), it is very likely from a statistical standpoint.



**Figure 3 - Completion Date Confidence Graph**

The third output of EBS is a graph of possible completion dates for each developer (see Figure 4 - Possible Completion Dates by Developer). This graph is developed by displaying the range of possible completion dates for each developer on the horizontal axis. Developers with a wide range of possible completion dates can be identified and potentially trained on improving their estimating skills. In addition, project managers can identify resources along the critical path that may be delaying the project, due to factors such as a high workload.



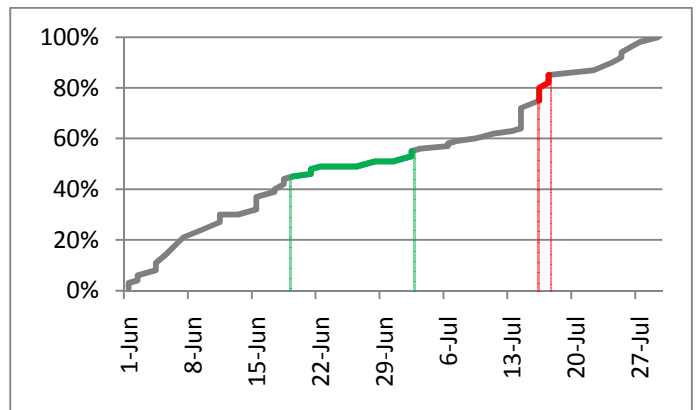
**Figure 4 - Possible Completion Dates by Developer**

**Strengths of EBS**

Evidence Based Scheduling demonstrates several obvious strengths, including simplicity, readability, and graceful degradation. Some less obvious benefits include the inherent difficulty in “gaming” the system, simplistic handling of vacation and holidays, and ease of handling of fractionally assigned resources.

At its core, EBS estimates are based simply on expert opinion, with historical adjustments. When comparing EBS to other common schedule estimating techniques, such as *function point analysis*, it is clear that EBS requires much less complexity and sophistication. For example, function point analysis requires in-depth analysis of program features to produce a function point estimate which is then adjusted and converted to a rough size estimate. EBS, on the other hand, simply requires assigning tasks, asking the developer how long the task is expected to take and then adjusting for historical accuracy.

Another aspect of the simplicity of EBS is the measure of confidence in a particular estimate. Rather than calculate a confidence value for a particular date, the reader simply notes the slope of the curve surrounding the date. If the slope is high, then confidence is high; if the slope is low, confidence is low. Consider the scenario represented in Figure 5 - Confidence and Slope. The probability of completing construction around Jun 28<sup>th</sup> is 50%. However, the range of dates falling between 45% and 55% cover a two week period. On the other hand, the range of dates falling from 75% to 85% covers only two days. Therefore, there is high confidence in the 80% probability, but low confidence in the 50% probability.



**Figure 5 - Confidence and Slope**

EBS degrades very gracefully with respect to poor estimates. In other words, an estimator who has a horrible track record and has velocities that are all over the map will not ruin the quality of the EBS output. Their input will simply widen the range of probable dates proportional to their contribution to the project.

EBS requires developers to track their time at the task level, but then goes further by instructing them to ignore interruptions, breaks, etc. Since these distractions are a natural part of the developer's environment, they should be maintained as part of the estimation process. Therefore, actual interruptions during the gathering of historical data will provide insight into probable interruptions during later projects. This concept can be extended to include vacation, sick days, holidays, and fractionally assigned resources. For example, traditional project management assumes resource allocations should be tracked and adjusted as resources request time off from work. EBS, on the other hand, suggests that these items are simply contributors to a developer's error in estimation accuracy.

A less obvious strength of EBS is the project manager's ability to analyze dates slippages. Using the Completion Date Confidence Graph (see Figure 3), if all three lines on the graph are rising, then the completion date is slipping into the future. If the completion date is moving into the future by faster than one day per day, then more work is being added each day than is being completed. The project manager can review the workload to determine if this is resulting from the addition of work or from slower than expected progress.

#### *Weaknesses of EBS*

While EBS appears to be suitable for the environment and resources of Fog Creek Software, it may not be appropriate everywhere. Some organizations may find specific aspects of EBS, such as the gathering of historical data, problematic. Other organizations may find EBS too lightweight and may require additional structure not provided by the tool.

#### **Scaling**

The most obvious weakness of EBS is its inability to scale when projects grow or shrink beyond the scope of the historical data. EBS assumes a direct correlation between historical data and future performance. This assumption may be accurate for projects of similar composition, including scope, size, and duration. However, when a future project does not fall within roughly the same parameters as the historical projects, the correlation is lost. Research has shown for software projects that "the coordination effort increases as  $n(n-1)/2$ , where  $n$  shows the number of parts."<sup>6</sup> EBS does not account for this increased project complexity.

#### **Gathering Historical Data**

The second weakness of EBS lies with the effort in gathering historical data. Small, technical organizations, like Fog Creek, which have strong technical leadership may be able to accurately and consistently gather historical data on an ongoing basis. However, not all organizations are as focused and well-managed as Fog Creek. Many factors in an organization can negatively affect the gathering of historical data. For example, an organization with high turnover, either at the company level or between departments, will find it difficult to accumulate a significant number of data points for all developers. Additionally, an organization must place a high value on the gathering of accurate data to properly encour-

age developers to actively participate and devote the required effort. Unfortunately, not all organizations are this focused in terms of goals or strategy.

#### **Buffer Time**

Another weakness of EBS is its reliance of "buffer time" to handle scope creep, integration, and testing. EBS expects the project manager to add an arbitrary amount of buffer time to the project. This buffer time will be consumed by unplanned activities, such as new features and integration testing. Of course, a traditional project plan will allocate specific tasks and durations for items such as code integration and testing. Without these items in the WBS, several critical activities are lost for both planning purposes and historical data analysis.

#### **Lack of Tools for the Individual**

A less obvious weakness of EBS is that it does not provide individual developers with tools for building estimates or improving their estimation skills. Developers utilize a "gut feel" approach to producing estimates, which may be suitable for tasks which are repetitions of previous work, but provides no basis for tasks involving new problems.

#### **Lack of Consistency across the Organization**

One of the fundamental principles of size estimation is that it must be empirical, repeatable, and consistent. EBS is definitely empirical, as it provides a clear estimation of confidence in dates, which can translate directly to a measure of estimation error. EBS is repeatable since the estimates are built on historical data and, in theory, should improve over time and usage. However, EBS cannot guarantee consistency across an organization, since developers use a "gut feel" approach. Without more formal methods, the "gut feel" estimate may vary based on a variety of factors. In other words, Monday's guess may be different from Friday's guess, simply due to the developer's emotional state. In addition, while EBS may account for an individual's accuracy in their estimates, it does not provide tools to improve organizational estimates as a whole.

#### *Other uses of EBS*

EBS can be used as a tool for relative measurements of an individual developer's estimation skills. In essence, the velocities tracked by EBS for a particular developer may be compared to other developers and to the organization as a whole. This information may be used to target specific developers for additional training or guidance.

Project managers can also inspect the accuracy probabilities of developers on the critical path. This helps the project manager identify which tasks may need to be reassigned to other developers with more reliable estimation skills or the project manager may simply keep a closer watch on high risk activities.

#### **Role of Scheduling in a Software Process**

In a typical software process, such as PCSE, the scheduling step follows identification of *features*, production of the *conceptual design*, *feature set* selection, identification of *tasks*, and initial estimation of *task durations* (see Figure 6 - PCSE Process). Scheduling is followed by the completion of the *architectural design*, *construction*, and *refactoring*. As such, there are no obstacles to implementing EBS in a traditional software process since it does not require modifying the process outside the scheduling phase,

other than to gather historical data.

At a minimum, EBS creates a structured framework around an “expert opinion”-style estimation process, used by many organizations. Such a framework may simply serve as guide to interpreting these “seat of the pants” estimates or it may become an integral part of the estimation process. Regardless, proper scheduling is a key component of managing any software project.

Software projects with accurate scheduling estimates will typically have the necessary resource allocations and will contain few surprises, in terms of duration. On the other hand, underestimating a project may lead to cost overruns, overtime for resources, and unhappy clients. Overestimating a project's effort may lead to improper resource allocations or, in extreme cases, project cancellations.

### EBS for the Individual or Small Team (PCSE)

Practice-Centered Software Engineering (PCSE) is a minimalist approach to the Personal Software Process (PSP). PCSE is tailored to individual software engineers or small teams. As such, EBS should easily integrate into the scheduling and estimation portion of PCSE.

#### Challenges

Since PCSE is typically implemented for individuals or small groups, there may be a lack of historical data. Without a good base of historical data to work from, EBS must assume poor estimation skills, which results in lower-than-actual probabilities (confidence). This should not have a detrimental effect on the team's estimates and will eventually correct itself over time.

Similarly, small groups tend to work on smaller projects. Smaller projects, with a smaller WBS, will see less error correction across the project as poor estimates in different directions cancel each other out. For example, a project with hundreds of tasks will be minimally affected by one or two badly estimated tasks. However, a project with only twenty tasks will be severely affected by a few bad estimates.

#### Implementation Concerns

Implementing EBS in an organization utilizing PCSE requires addressing a few specific concerns. First, historical data may be in short supply, so a generic profile should be constructed for individuals lacking such data. The generic profile should error on the side of poor estimating skills, which will tend to decrease confidence in delivery dates. As tasks are completed, individual developer's data may be fed back into the EBS model at shorter intervals to more quickly develop a personal model for that particular developer.

Second, a consistent process for developing conceptual models and initial schedules should be constructed. Small teams tend to

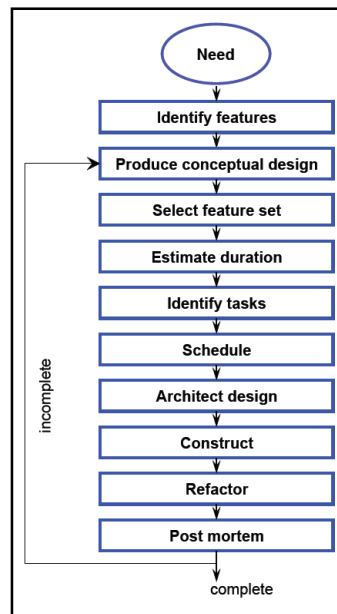


Figure 6 - PCSE Process

have less formal roles and these steps should not vary from one project to another due to individual styles. Therefore, a (semi-formal) process model should be used to ensure some level of uniformity.

Lastly, the size of the Monte Carlo simulation set should be tailored to produce a clear trending model. Sets which are too small will produce a more erratic graph. Sets above a certain size will produce no additional useful information and will slow processing and rendering of the data.

### Conclusions

Implementing Evidence Based Scheduling represents a strong step forward for organizations lacking a formal schedule estimation process. The simplicity and ease of use of EBS can allow for easy adoption, while placing a rudimentary structure around a typically unstructured process. Of course, the decision to implement EBS must be tempered with an introspective look at the organization. Not all organizations will find EBS compatible with their structure and culture, and those that do must recognize that it is no silver bullet.

### References

- [1] Fog Creek Software – About the Company. July 2009. Fog Creek Software. Accessed October 9, 2009. <<http://www.fogcreek.com/About.html>>
- [2] Monte Carlo Method – Wikipedia, the Free Encyclopedia. October 16, 2009. Wikimedia Foundation, Inc. Accessed October 19, 2009. <[http://en.wikipedia.org/wiki/Monte\\_Carlo\\_method](http://en.wikipedia.org/wiki/Monte_Carlo_method)>
- [3] Better Software Magazine. March 2007. StickyMinds.com. Accessed October 6, 2009. <<http://www.stickyminds.com/BetterSoftware/magazine.asp?fn=cifea&id=94>>
- [4] Evidence Based Scheduling. October 26, 2007. Fog Creek Software. Accessed October 6, 2009. <<http://www.joelonsoftware.com/items/2007/10/26.html>>
- [5] Agresti, William, et al, Manager's Handbook for Software Development, Revision 1, November 1990, page 3-2.
- [6] The Challenge of Large-Scale IT Projects . Ahmet Denker . World Academy of Science, Engineering and Technology 9 2005 , pages 24-27.