A Project Management Approach to Using Simulation for Cost Estimation on Large, Complex Software Development Projects

Abstract: It is very difficult for project managers to develop accurate cost and schedule estimates for large, complex software development projects. None of the approaches or tools available today can estimate the true cost of software with any high degree of accuracy early in a project. This paper provides an approach that utilizes a software development process simulation model that considers and conveys the level of uncertainty that exists when developing an initial estimate. A NASA project will be analyzed using simulation and data from the Software Engineering Laboratory to show the benefits of such an approach.

Keywords: Software Development Cost and Schedule Estimation, Simulation, Software Development Process Model

About the Authors

Carolyn Mizell is a NASA employee at the Kennedy Space Center where she has served as a project manager for over 10 years. She is presently working on her PhD in Industrial Engineering at the University of Central Florida.

Contact: Carolyn A. Mizell, NASA EA-C, Kennedy Space Center, FL 32899; phone: 321-867-8814; Carolyn.A.Mizell@nasa.gov

Linda Malone is a professor at the University of Central Florida. She is a Fellow of the American Statistical Association.

Cost and schedule estimation for large software development projects is historically inaccurate. Popular estimating models have been shown to be only within 25% of actual costs for 50% of the time (Ferens and Christensen 1998). Using a simulation tool, this paper presents an estimation approach that illustrates the effects of normal working dynamics on the cost of a large software development project throughout the project's evolution. Simulation models can be used to communicate the uncertainty and complexity of the development process and can provide a check on other estimating methods that may be used.

The ability to obtain an accurate estimate of an entire project prior to its start is unfortunately unrealistic. And yet, cost and schedule estimates are necessary as management commits to funding such projects or bidding on a job. Simulation models are typically used to analyze the effects of process changes, and not for developing initial cost and schedule estimations. This paper will describe how simulation models can be used for this purpose and will show the benefits that can be obtained by using simulation as an estimation tool. The tool will illustrate the difficulties management faces in forecasting budgets at the beginning of a project and may encourage more realistic approaches to budgetary planning including phased funding.

The task of cost estimation for project managers of software development projects becomes more and more daunting as the size and complexity of the project increases. Complex software development projects are likened to pure research and development projects, with all of the inherent difficulties of managing and planning for work that is innovative and unique and that has uncertain requirements (Abdel-Hamid and Madnick 1991).

Planning purposes require that an estimate be developed at a time in the project when the values of key parameters such as product size and staff capabilities are unknown. This makes it unrealistic to provide an accurate estimate. Even if the values of key variables could be known with certainty at the beginning of a project, software activities are labor intensive and prone to all the complex and dynamic factors which affect human performance. Therefore, software development is not a deterministic activity and an estimate will require adjustments throughout the project until all the variables are known.

The initial estimate for a project is the most difficult and least accurate since there is less data available. Different tools and techniques for developing software project estimates exist, but none are guaranteed to give an accurate estimate. Often, though, the initial (and highly uncertain) estimate becomes the official estimate for the entire project and is used to judge whether or not the project is successful.

Trying to obtain a precise estimate at a very early stage in a project has lead to the use of techniques that do not depict uncertainty and complexity of the factors. Human nature prefers a single number for an estimate as opposed to a range of numbers, even though a range estimate will have a much higher chance of including an accurate value (Boehm and Fairly 2000). Much of the research work carried out in the software cost estimation field has been devoted to algorithmic models such as COCOMO and yet, methods that rely on expert judgment are still the most commonly used approaches (Agarwal and Kumar 2001).

Expert judgment approaches rely on experience on past projects and published industry averages. Average data does not tell the whole story and although past projects may seem similar, they will not have the same development costs, since estimates based on past experience do not account for changes in environments, politics, or organizations (Abdel-Hamid and Madnick 1991). In addition, historical data and experts' memories of the past can be tainted. Even though expert judgment is the most often used technique, empirical software estimation models such as COCOMO are still widely used. These tools provide rigor to the estimating process, but the portability of these tools to environments different from that for which the tool was developed comes into question. The developer of COCOMO, Barry Boehm, admits that COCOMO is not right for every development environment (Boehm, Abts et al. 2000).

In essence, none of the approaches or tools available today can estimate the true cost of software with any high degree of accuracy early in a project. Managers should be presented with a technique that identifies risk and uncertainty based on the seemingly random nature of the variables and the complexity of the project system. Although managers must commit to a budget number and schedule, they should not be given a false sense of confidence in a point estimate. Adequate management reserves and phased funding should be considered to account for uncertainty, especially for larger and more complex software development projects.

Baseline Simulation Model

Simulation models have been used to derive implications about the behavior of an organization through integration of the multiple functions of the software development process (Abdel-Hamid and Madnick 1991) and to quantitatively evaluate the performance of alternative software processes and process changes (Raffo 1996). Simulation is typically used for these types of analyses after a project is underway or completed. This paper will demonstrate the benefits of using a simulation model for early cost estimation that is done before a project begins or very early into a project when uncertainty about key parameters is very high.

The Process Analysis Tradeoff Tool, PATT ©, is a discrete event process simulation model that was developed for NASA to assess the benefits of Independent Verification and Validation (IV&V) on the IEEE 12207 software development process (Raffo and Wakeland 2003). The tool is intended to enable adaptation to multiple projects and IV&V techniques. The model uses industry average data for input variables such as product size, productivity (LOC/Hr), and defects (per KSLOC). The user provides the per cent of overall effort that should be allocated to each process step as well as the number of desired staff for each step. The model outputs the size, effort, rework effort, entire process duration, average duration, number of injected defects, detected defects, and corrected defects.

Model Input Data

The use of probability distributions for key variables such as size, productivity, and defects is a truer model of reality, especially in the early stages of a project. The model's outcomes will be driven by random variables drawn from the probability distributions. Numerous runs of the process with different random numbers will provide the most meaningful information and will allow for the calculation of confidence intervals for each quantity of interest.

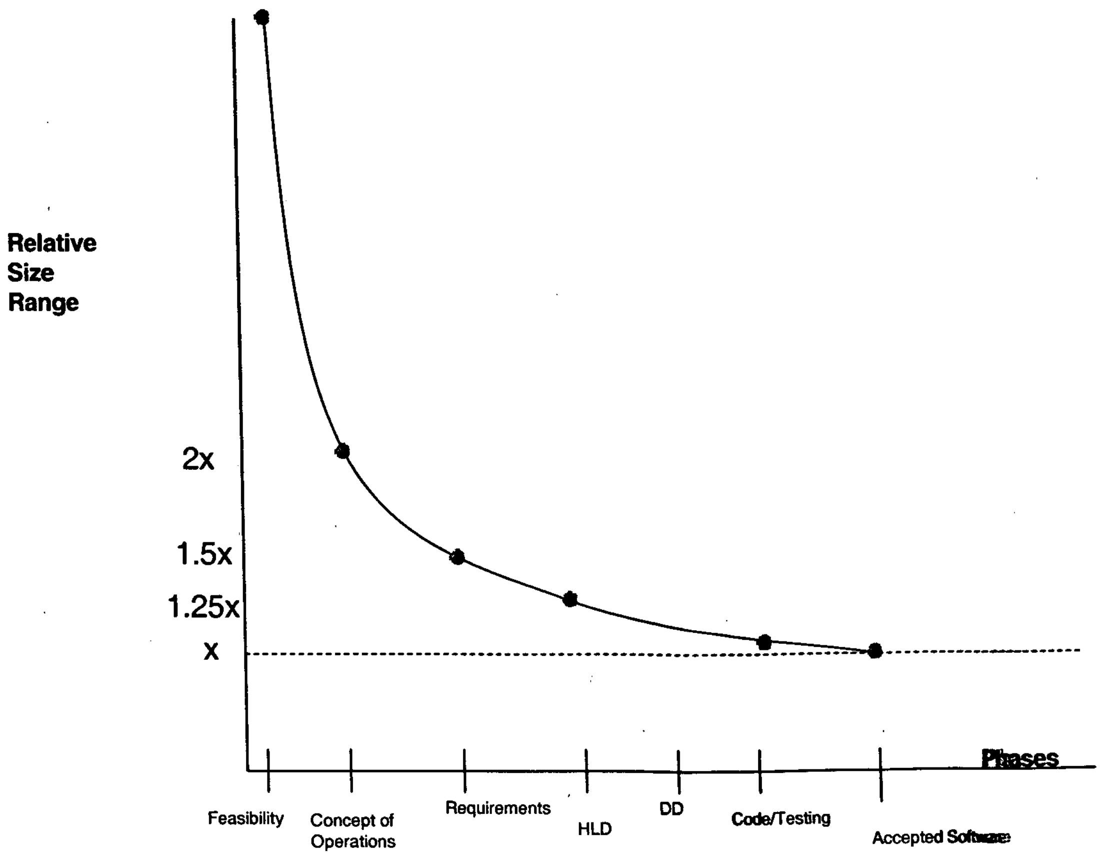
The Software Engineering Laboratory (SEL) began collecting data for NASA flight software development projects at Goddard Space Flight Center in 1976 and served as a major resource in software process improvement activities. (Basili, McGarry et al. 2002). Extensive project and product data was collected for over 200 projects and is available to the public (SEL 1997). It is recognized that the collection, analysis and retention of historical data for software development needs to be increased (Fairley 1992), but the process of doing so is complex, costly and time-consuming. Therefore, many organizations do not have adequate amounts of reliable data at their disposal. Data from the SEL have served as the basis of many software development "rules of thumb" concerning lifecycle activities and defect generation activities. However, this data has not been used for developing appropriate probability distributions for use in stochastic simulation. The use of the SEL historical data for cost estimation using stochastic simulation should provide benefit for early estimation in similar environments. The SEL data is especially appropriate for projects developed in the NASA environment with its stringent testing and reliability requirements.

As mentioned, the entire set of project data collected by the SEL is available. For the purposes of this research, subsets of the data that were collected and organized by the NSF Center for Empirically Based Software Engineering (CeBASE 2005) were used. Probability distributions were fitted for productivity data and defect injection data.

The uncertainty that exists in estimating the size of the final software product is also very important since size has been shown to be the key factor, followed by the effort adjustment multipliers, in models such as COCOMO (Musilek and Pedrycz 2002). Experts claim that the problem of how to accurately size software is the greatest roadblock to improving estimation (Boehm 1981). The problems of incomplete, vague requirements as well as requirements creep are almost always present for software development projects and will cause the size of the project to change over its development life cycle.

In order to capture the impact and uncertainty of sizing a software development project, a uniform distribution is used for the estimated size of the product. The he figure in Exhibit 1 is adapted from a figure given in Software Cost Estimation with Cocomo II and portrays how much size can be underestimated at different points in the project lifecycle. For example, at the very early stage of a project where there is a concept of operations but no firm requirements,, a size estimate can be off by a factor of 2. This means that an estimate at this point in the project of 1 Million LOC could actually end up being as high as 2 Million LOC by the end of the project. Therefore, using the figure in Exhibit 1 as a guide for size estimates at specific lifecycle phase such as concept of operations, a size estimate of 1 Million LOC would equate to the following distribution: Uniform (1 Million, 2 Million). The parameters of the distribution will change based on the estimate and the phase of the lifecycle for which the estimate is developed.

Exhibit 1: Size Uncertainty According to Lifecycle Phase (Adapted from Boehm 2000)



NASA Project

NASA's software development projects are often large and complex due to the mission critical nature of their business. With human life and billion dollar payloads at risk, the NASA development environment is more stringent and complex than that of many other software development industries. For example, a bug in a windows product can cost money and customer dissatisfaction, but an error in launch processing software can lead to loss of life and vehicle.

NASA faces all of the same estimating difficulties discussed thus far. It is not uncommon to have managers who underestimate costs and then must continually request new funding at the risk of project cancellation. In fact, the U.S. General Accounting Office has criticized NASA managers for the past decade for failing to create realistic budgets for new projects. A recent government watchdog analysis showed that a majority of 27 recent projects was found to have costs that were very different from the initial estimates, some by as much as 94% (Asaravala 2004).

In order to explore the benefits of using simulation for cost estimation on large, complex software development projects, a real NASA project will be used as the subject of this study. The project was cancelled after seven years and experienced cost overruns and schedule slippages throughout its history. The project faced many of the following common problems that make software development cost estimation especially difficult: costs and schedules are pre-determined by an outside source, the software development process is not fully understood or analyzed, requirements are not well-defined and prone to changes, new projects are almost always different from past ones, and software practitioners do not collect enough data for past projects (Agarwal and Kumar 2001).

Different estimating techniques were used to develop estimates for the project. This paper will focus on the project's early estimates and will outline how the simulation tool could have been used during this timeframe as well as throughout the project.

The Approach

The information known at the time of the early estimates will be used with the simulation approach to analyze the benefits of such an approach.

1. Utilize Software Development Process Model

A graphical representation of a software development lifecycle process is useful when educating decision makers on the inherent complexity which makes large, complex software development projects so difficult. The model allows for the user to understand the flow of work that needs to be accomplished and to analyze the impact of things such as rework through graphic display of the software product moving through the lifecycle steps.

2. Capture Uncertainty for Key Parameters by Using Probability Distributions

Three key parameters that greatly affect the cost and schedule of a software development project are size of the product, productivity of project personnel, and defect rates. It is possible to find average values for productivity and defect rates in the literature, but it is important to point out that these parameters are subject to many influences that can greatly vary the value of the parameters throughout the course of a project. A great deal of uncertainty exists when trying to estimate values for these very important parameters before a project begins and early in the project. Therefore, the use of probability distributions will allow for developing range estimates considering the uncertainty that exists before a project begins.

It is desirable to have data from similar projects and environments for developing data distributions in order to obtain credibility for using the data as inputs to the simulation model. If adequate data is not available, the parameters for reasonable distributions can be estimated.

3. Run Model and Obtain Confidence Intervals for Effort and Schedule

Calculate confidence intervals for the two primary estimation parameters of effort and duration. When presenting the data, the top half of the confidence interval should be focused on in order to deter the desire to accept the lower bound of the interval in order to meet the lowest possible cost and schedule. This will take into account other factors that may not be accounted for and will remind decision makers that all will never go as planned and that Murphy's Law is alive and well when it comes to large, software development projects.

4. Compare Model Results with Other Estimating Techniques

Ideally, estimates developed with other techniques should fall within the upper half-width of the confidence interval in order to allow for those unplanned and unfavorable events that will occur during the course of every big project.

5. Use Model Results to Debate Unrealistic Budgets

If a budget is set with values that do not fall within the confidence intervals produced by the simulation model, then the model should be used as a tool for debate. The key input parameters can be varied to show the impacts of such variation. The model should be run with animation so that decision makers can visualize the process and reasons for its complexity.

6. Update Model with Actual Project Data as Project Evolves

Actual project data can be used as inputs to the model as the project evolves. The model can be used to analyze effort and schedule to complete based on actual project values to date. The model can also be used to study problem areas and the effects of potential solutions.

Approach Applied to Project:

The NASA project documentation shows that there were two official estimates before the project started. The first was an estimate by analogy and the second a bottoms-up estimate. Both of these estimates would officially result in the same total cost and schedule for the project. A review of interim bottoms-up estimate documents and personal notes show that the total project cost values were lowered for the final and official estimate. Even at this very early point in the project, there was great pressure to develop estimates that matched an acceptable budget and schedule. The budget was set for a cost of approximately 1400 labor years of effort and five years for the schedule.

1. Baseline Model

The earliest estimated size for the project was 1.4 million lines of code and this was based on a rough analogy with previous projects. At this point in time, no decisions had been made on an acceptable concept of operations, architecture, or lifecycle process for the project. Previous similar projects had used a structured waterfall process so it is reasonable to utilize the baseline PATT discrete event process model.

2. Data Distributions

Since there is no actual project data at this point, data from the Software Engineering Laboratory will be used to develop appropriate distributions for use in the model.

Size: Referring to Exhibit 1, the size estimate can be underestimated by up to a factor of 4 at this very early point in the lifecycle. Therefore, a uniform distribution with parameters 1.4M and 5.6M will be used in the simulation. Note: An interesting fact to point out is that the estimated size of the project at the time it was cancelled was 5.8M LOC.

Productivity: Productivity data for over 140 projects from the Software Engineering Laboratory was analyzed and fitted with distributions using "best fit" software. The following distribution was selected for productivity: Erlang (1.36, 3). For different environments where there is not adequate historical data, a triangular distribution could be used and the parameters approximated by considering the minimum, maximum, and most likely values.

Defect Rates: Values for defect insertion rates that range between 10 and 60 defects per thousand lines can be found in the literature (CeBASE 2004) with the smaller rates observed for projects that utilize disciplined and structured software engineering practices. The SEL data was used to analyze the number of defects inserted per thousand lines of code for different phases in the lifecycle. The PATT model is set up to accept six different types of defects, three of which relate to the lifecycle phases of requirements, design, coding, and testing. The other two types of possible errors are from bad fixes and documentation errors. Four distributions were developed for the phases of requirements, design, coding and testing and are provided in Exhibit 2.

Exhibit 2: Defect Probability Distributions by Phase

Phase	**Distribution**
Requirements	**Lognormal (2.62, 7.1)**
Design	**Lognormal (17.13, 73.35)**
Coding	**Weibull (28.39, 0.81)**
Testing	**Exponential (40.9)**

A nominal value of 30 defects per thousand lines was used for bad fixes and documentation errors, since this type of defect data was not available for distribution fitting.

For development environments where there is not adequate data to fit distributions and for which the SEL data is not appropriate, lognormal distributions can be used. Previous work demonstrates that a lognormal distribution for defect injection rates is appropriate in software process simulation modeling (Raffo 1996). Industry averages can be used for selecting parameters of the distribution.

3. Run Model.

The model was run for five replications with the following results for size, productivity, and defect injection rates:

Based on the initial run set of five replications, the 95% confidence intervals for effort and schedule are:

Effort:	**16,705+/-7536.2 Labor Months**
Duration:	**68.1 +/- 37.24 Months**

These are not very useful confidence intervals since the half widths are too large.

In order to obtain a smaller half-width, the model needs to have a substantially higher number of replications.

With the desire of obtaining less than 10% error on both parameters, the number of replications is solved for by the following:

Effort:

$$n \cong 5\left[7536^2 / 1670^2 \right] \equiv 101.8$$

Duration:

$$n \cong 5\left[37^2 / 7^2 \right] \equiv 140$$

Therefore, run the model for 150 replications and calculate confidence intervals.

Exhibit 3 provides the PATT main screen table of results:

Exhibit 3: Results for PATT Model Run of 150 Replications

Metric	Mean	S.D.	Units
Size	4812.8	55	KSLOC
Effort	21040.5	1048.2	Person Months
Rework Effort	1682.2	129	Person Months
Duration	91.7	11.2	Months
Avg. Duration	73.9	9.5	Months
Inj. Defects	103277.3	3582	
Det. Defects	94093.6	3149	
Cor. Defects	93866.7	3144	
Latent Defects	9410.6	526	

Exhibit 4 summarizes the results of a run set with 150 replications:

Exhibit 4: Effort and Duration Confidence Intervals

	Effort	Duration
Mean	1416 Labor Years	6.25 Years
Half-Width	65 Labor Years	0.32 Years
Confidence Interval	[1351, 1481] Labor Years	[5.93, 6.57] Years
Presented Confidence Interval (Upper Half)	**[1416, 1481] Labor Years**	**[6.25, 6.57] Years**

4. Compare Results with Other Estimating Techniques

The estimate by analogy led to a budget of 1400 labor years and 5 calendar years. The simulation model produces values that demonstrate that there is substantial risk in accepting this budget. The model can be used to show that many factors must be considered and that these add to the complexity, time and cost for the project. For instance, decision makers need to be made aware of the impacts of defects and rework on the project, which can be substantial. The rework effort calculated using the simulation model is 8% of the overall effort. Also, the size of the project and the productivity of workers that have not yet been assembled cannot be estimated with any degree of accuracy. The model tries to account for this complexity and uncertainty. The visual display that lays out the waterfall process also adds value in considering aspects that are important and that can be of great impact.

5. Debate Unrealistic Budgets

The first area of concern should be the schedule. Fred Brooks states in his famous book (Brooks 1978) that more software development projects fail due to a lack of calendar time than all other factors combined. Capers Jones states that, "Once a project blindly lurches toward an impossible delivery date, the rest of the disaster will occur almost inevitably" (Jones 1998).

One of the top ten software management tenets states that software development schedules should not be compressed by more than 25% of nominal (Royce 1998). The initial schedule of five years is dangerously close to a 25% compression of the time estimate from the simulation model. Therefore, decision makers need to be made aware that a five year schedule is very risky and that a longer development schedule should be given serious consideration and its impacts assessed at this very early point since NASA is always concerned with launch manifests being affected by delivery dates.

The estimate of 1400 labor years falls a little short of the values in the confidence interval. Management should be made aware that factors such as productivity and size can greatly impact this estimate. At this point in time, the size of the product will most likely vary from the estimated 1.4 Million LOC. The productivity distribution is based on a NASA development environment of cohesive development teams with low personnel turnover and in some cases, high reuse and development language advantages. It is appropriate to accept these conditions for this project's development environment at this early point, but the environment could easily be different and these types of differences could substantially affect the productivity numbers. The benefit to closely analyzing a very early budget and emphasizing obvious risks such as schedule in this case is that decision makers do not become too comfortable with unrealistic estimates. The longer unrealistic numbers are considered as acceptable, the harder it becomes to change those numbers. Also, changes to budgets later rather than earlier (preferably before a project starts) more negatively affect the team's reputation and management's confidence in the ability to successfully complete the project.

Second Estimate:

The objective of the Bottoms-Up Estimate was to develop a rough order of magnitude cost and schedule for a five year project. As part of this estimate, the system architecture was selected and a concept of operations was developed.

1. Baseline Model

The size of the software was estimated to be 3.8 Million LOC. A productivity rate of 1 LOC/Hr was used for the estimate. The bottoms-up estimate adheres to the original schedule of five years. The estimate summary states that the schedule is aggressive and success-driven, but does not recommend a longer schedule.

2. Data Distributions

There will be changes in the size and productivity distributions for this estimate based on what was known at this point in the project. The defect distributions will be unchanged from the values used for the first estimate. A Uniform Distribution with parameters 3.8M and 7.6M will be used for size. A triangular distribution with parameters of 0.5 for minimum, 3 for maximum, and 1 for most likely will be utilized for one run set and then compared to a second run set with the original productivity distribution of

Erlang (1.36, 3).

3. Run Model

The model was run two times with 150 replications for each. Exhibit 5 summarizes data from the runs:

Exhibit 5: Summary Data for Multiple Run Sets with Different Productivity Distributions

	Input Size	Input Productivity	Effort Confidence Interval (Upper Half)	Duration Confidence Interval (Upper Half)
Run Set #1	Uniform (3800,7600)	Erlang (1.36,3)	[2322, 2384]	[10.25, 10.6]
Run Set #2	Uniform (3800,7600)	Triangular (0.5,1,3)	[2028, 2080]	[8.7, 8.9]

4. Compare with other Estimating Techniques

The data from these two run sets should clearly send up a warning signal about accepting the established budget of 1400 labor years of effort and five years of schedule. The simulation model demonstrates the impact of the 3.8 million LOC size estimate and productivity estimate of 1 LOC/Hr. The established budget represents an unrealistic goal that is setting the project up for failure.

5. Debate Unrealistic Budgets

This approach would have allowed project members to raise concern over the established and unrealistic budget from the initial time it was presented. This second run of the model in conjunction with the detailed bottoms-up should add validity to describing the predetermined budget as impractical. Even if decision makers agree to an aggressive schedule and budget, the simulation model can be used to bound the aggressiveness and to instruct on reasons for concern.

Summary and Conclusions:

Simulation process models can provide benefit to the estimation process for software development projects, although they are not typically used for this purpose. A simulation tool can be used to graphically portray the complexity of the development process and can be used to explore the effects of uncertainty in the key parameters of size, productivity, and defects. Simulation process models can be used in conjunction with other popular estimating methods to serve as a check on the validity of the estimate developed with these other techniques. It is necessary to develop estimates before a project begins, but it is also necessary to understand that a point estimate developed with many unknowns and uncertainty is not going to be accurate. There is the danger that unrealistic cost and schedule estimates agreed to in order to get a project started can become the official budget and schedule with no easy way of revisiting and changing them. The goal of this work has been to demonstrate the benefits of using a simulation model when estimating to allow for more realistic budget and schedule determination including an interval estimate to help focus on the uncertainty of the estimates.

Acknowledgements:

The author would like to thank Dr. David Raffo and Portand State University for providing the Process Analysis Tradeoff Tool for this work.

References

Abdel-Hamid, T. K. and S. E. Madnick (1991). Software Project Dynamics: An Integrated Approach. Englewood Cliffs, Prentice Hall.

Agarwal, R. and M. Kumar (2001). "Estimating software projects." Software Engineerinng Notes 26(4): 60-67.

Asaravala, A. (2004). Cost Estimation: Learning from NASA's Troubles. Software Development.

Basili, V. R., F. E. McGarry, et al. (2002). Lessons learned from 25 years of process improvement: The Rise and Fall of the NASA Software Engineering Laboratory. ICSE, Orlando, FL.

Boehm, B. W. (1981). Software Engineering Economics. Englewood Cliffs, Prentice Hall.

Boehm, B. W., C. Abts, et al. (2000). Software Cost Estimation with COCOMO II. Upper Saddle River, Prentice Hall PTR.

Boehm, B. W. and R. E. Fairly (2000). "Software Estimation Perspectives." IEEE Software 17(6): 22-26.

Brooks, F. P. (1978). The Mythical Man-Month: Essays on Software Engineering. Reading, Addison-Wesley Publishing Company.

CeBASE. (2004). "eWorkshop on Software Inspections and Pair Programming." from www.cebase.org.

CeBASE. (2005). "Software Engineering Laboratory Data." from www.cebase.org.

Fairley, R. E. (1992). Recent Advances in Software Estimation Techniques. Proceedings of the 14th international conference on Software engineering, Melbourne, Australia.

Ferens, D. V. and D. S. Christensen (1998). Calibrating Software Cost Models to Department of Defense Databases - A Review of Ten Studies. ISPA/SCEA Joint International Conference Proceedings. Toronto, Canada, Air Force Institute of Technology: 15.

Jones, C. (1998). Estimating Software Costs. New York, McGraw-Hill.

Musilek, P. and W. Pedrycz (2002). On the sensitivity of COCOMO II software cost estimation model. Eighth IEEE Symposium on Software Metrics.

Raffo, D. (1996). Modeling software process quantitatively and assessing the impact of potential process changes on process performance. Industrial Administration. Pittsburgh, Carnegie Mellon University. **PhD:** 308.

Raffo, D. and W. Wakeland (2003). Assessing IV&V Benefits Using Simulation. 28th Annual NASA Goddard Software Engineering Workshop.

Royce, W. (1998). Software Project Management: A Unified Framework, Addison-Wesley.

SEL (1997). Guide to Software Engineering Laboratory Data Collection and Reporting. Software Engineering Laboratory Series. Greenbelt, Goddard Space Flight Center.