Project Plan: MSE Portfolio Project – Construction Phase

"Plans are nothing; planning is everything". –Dwight D. Eisenhower

September 17, 2010

Prepared by Doug Smith
Version 2.0

Table of Contents

Revision History

Version	Date	Changes
1.0	12/28/2009	First draft.
2.0	10/17/2010	Redone after initial architecture disproven, second architecture redone

Introduction

This is the project plan for completing the MSE Software Engineering Portfolio project, which is done in partial fulfillment of the Kansas State University (KSU) Master of Software Engineering degree. This documents the plan and cost model for producing the artifacts associated with the software engineering portfolio.

The plan in this document reflects the current state of the project, namely the completion of the elaboration phase of the project, transitioning to the construction phase.

Project Life-Cycle Model

The project life-cycle model for KSU portfolio projects is a slightly modified version of the Rational Unified Process. There are two main phases – engineering and production, with the engineering phase divided into the inception phase, followed by the elaboration phase. RUP divides the production phase into construction and transition phases. Since there is no installation of the system developed into a production environment, no user training or acceptance testing, there is no transition phase performed as part of portfolio projects.

The following diagram shows the 3 phases of the project, with a "you are here" indicator showing where in the project this version of the plan sits.

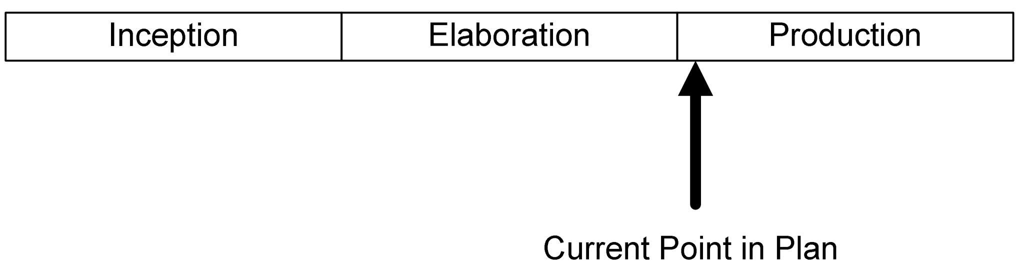


Figure 1 Point in Project Reflected in this Plan

The Plan

The artifacts that are required to be produced as part of the MSE Software Engineering portfolio are detailed on the KSU Computer and Information Sciences web site [1]. Based on the required artifacts, the following work breakdown structure was created for the construction phase.

ID		Task Name	Duration	Start	Finish	Predecessors
1		**Contruction Phase**	**33 days**	**Mon 09/20/10**	**Wed 11/03/10**	
2		**Finish Code**	**13 days**	**Mon 09/20/10**	**Wed 10/06/10**	
3		Create maven build	2 days	Mon 09/20/10	Tue 09/21/10	
4		Integrate sonar	1 day	Wed 09/22/10	Wed 09/22/10	3
5		Bring code into compliance with	5 days	Thu 09/23/10	Wed 09/29/10	4
6		Bring unit test coverage into com	5 days	Thu 09/30/10	Wed 10/06/10	5
7		Complete component design docume	5 days	Thu 10/07/10	Wed 10/13/10	2
8		**Complete Scale Testing**	**7 days**	**Thu 10/14/10**	**Fri 10/22/10**	**7**
9		Run Scale Tests	5 days	Thu 10/14/10	Wed 10/20/10	
10		Test Adding/Removing Cluster N	1 day	Thu 10/21/10	Thu 10/21/10	9
11		Write Up Test Assessment	1 day	Fri 10/22/10	Fri 10/22/10	10
12		Write User Manual	1 day	Mon 10/25/10	Mon 10/25/10	8
13		Complete Project Assessment	2 days	Tue 10/26/10	Wed 10/27/10	12
14		Complete Annotated Bibliography	3 days	Thu 10/28/10	Mon 11/01/10	13
15		Ensure Presentation 2 Action Items C	2 days	Tue 11/02/10	Wed 11/03/10	14

Figure 2 Work Breakdown Structure

The Gantt chart associated with the plan is available as an artifact on the associated project web site.

Scheduling Methodology

The scheduling methodology used to produce the project plan is completely ad hoc. The tasks to finish the project were enumerated in the work breakdown structure, and scheduled optimistically based on effort, not duration.

Over the life of this project the ability to devote time to it on a regular basis has been elusive, due to professional and family commitments.

Cost-Estimate

COCOMO II Model

For estimating cost, I have selected the COCOMO II model. This is an update to the COCOMO model that better reflects the realities of modern software development practices. This model features different cost predictors based on the current phase of the project. For this point in the project, I will use the Post Architecture model. Calculation of effort and schedule is detailed in [2] and [3].

The COCOMO II model calculates effort in person months using the following equation:

$$PM = A \times Size^{E} \times \prod_{i=1}^{n} EM_{i}$$

Where A = 2.94 for COCOMO II.2000

COCOMO II also provides a schedule estimation equation that gives the time to develop in calendar months:

$$TDEV = \left[C \times \left(PM_{NS}\right)^{(D+0.2\times(E-B))}\right] \times \frac{SCED\%}{100}$$

Where C=3.67, D=0.28, and B=0.91

E represents a scale factor to take into account potential economies of scale, where the size of the project can influence productivity gains or losses as the project is in flight. It is based on if the project undertaken by the development organization has precedence, and on other factors including development flexibility, risk resolution, team cohesion, and process maturity. I have selected nominal values for all these factors, as I believe there will be no scale effects in this project.

Project Sizing Methodology

For sizing estimates, the current size of the implementation based on the architectural proof of concept implementation and a projection of the remaining work was used to arrive at the project size used in the COCOMO II model. Specifically:

- To date, roughly 2600 SLOC has been produced.
- I estimate approximately 400 SLOC remains to be produced.

Note the SLOC counts have changed significantly from the size estimates used in the vision phase of the project. The SLOC counts used for the vision estimate were produced using the unix command 'wc –l' on proof of concept code base, then extrapolating, both on XML configuration files and on Java files. The SLOC used for this phase in the project is based on the SLOC produced by the SONAR tool

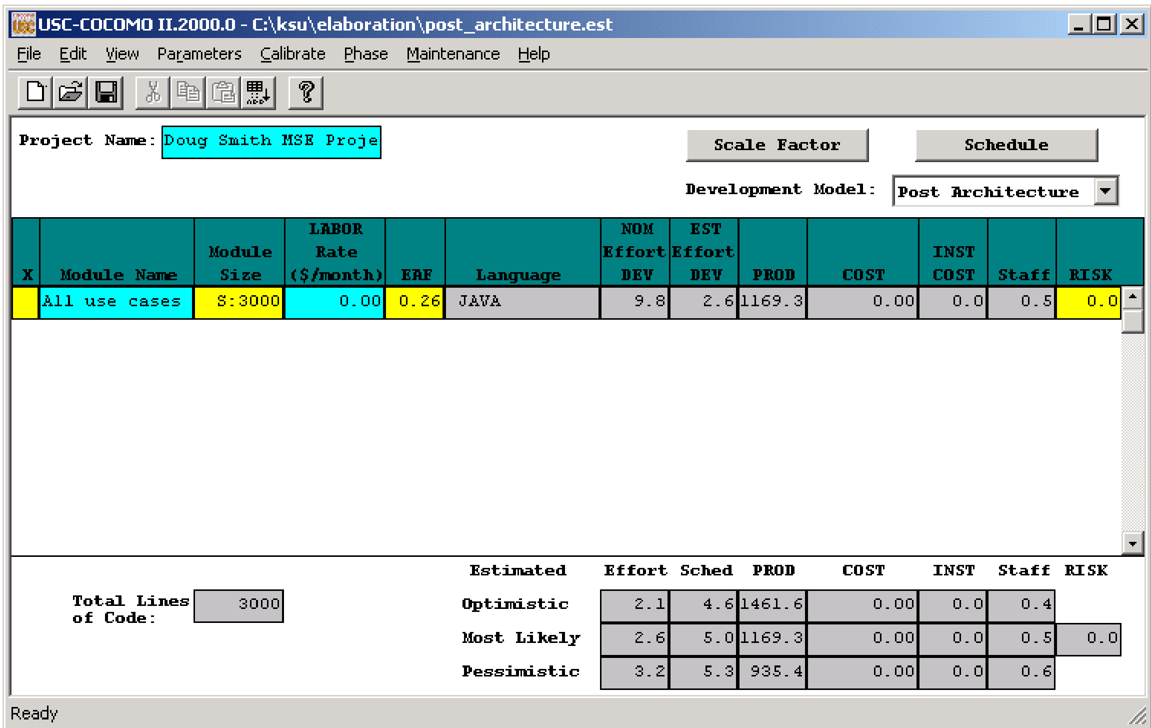
Based on the above an estimate of 3,000 SLOC is used for input into the model.

In producing the estimate, I had to select values for the components used to determine the effort adjustment factor, which the equation calculates as the product of the following effort multipliers (refer to [3] for a detailed explanation of each effort multiplier):

Component	Description	Selected Value and Comments
RELY	Required software reliability	*Nominal* (moderate, easily recoverable losses)
DATA	Database size in terms of the ration of the test database size to program SLOC	*Low*
CPLX	Product complexity	*Low*
RUSE	Developed for reuse	*Low*
DOCU	Documentation match to lifecycle	*Nominal*
TIME	Execution time constraints	*Nominal*
STOR	Main storage constraint	*Very Low*
PVOL	Platform volatility	*Very Low*
ACAP	Analyst capability	*Nominal* (cannot adjust this to influence effort, therefore nominal)
PCAP	Programmer capability	*Nominal*
PCON	Personel continuity	*Very High*
APEX	Applications experience	*Very High*
PLEX	Platform experience	*Very High*
LTEX	Language and tool experience	*Very High*

TOOL	Use of software tools	*Nominal* (nothing special above normal loosely integrated dev tools)
SITE	Fully colacted	*Very High*
SCED	required development schedule	*Nominal* (finished when I'm finished

For producing the COCOMO 2 estimates, I used the USC-COCOMOII software [4]. Plugging the above information into the tool yields the following results:



COCOMO II produces a range of values for effort and schedule. Looking at effort, the most likely estimate is given at 2.6 person months. As of 10/17/2009 I've expended 331 hours effort for the inception and elaboration phases, so according to COCOMO II (which defines a person month as 152 hours) there are 0.5 person months left or roughly 76 hours.

According to the project plan summarized above, I am anticipating completing the project with another 165 hours of effort (assume 5 hours per effort day in the project plan). Based on this there is a discrepancy between what the COCOMO II model projects, and what I project, even given the cone of uncertainty illustrated below (figure 3_. There are a few factors that can account for this difference:

- Working on the project in fits and starts reduces productivity, as opposed to applying a steady effort over the duration of the project.
- The size of the project is as the very low end of project sizes the model is targeted toward; the tool used for the estimates will not produce a value for SLOC less than 2000.
- Some of the contributing components to the effort adjustment factory could be incorrect.
- The estimates in the project plan will have varying degrees of accuracy.

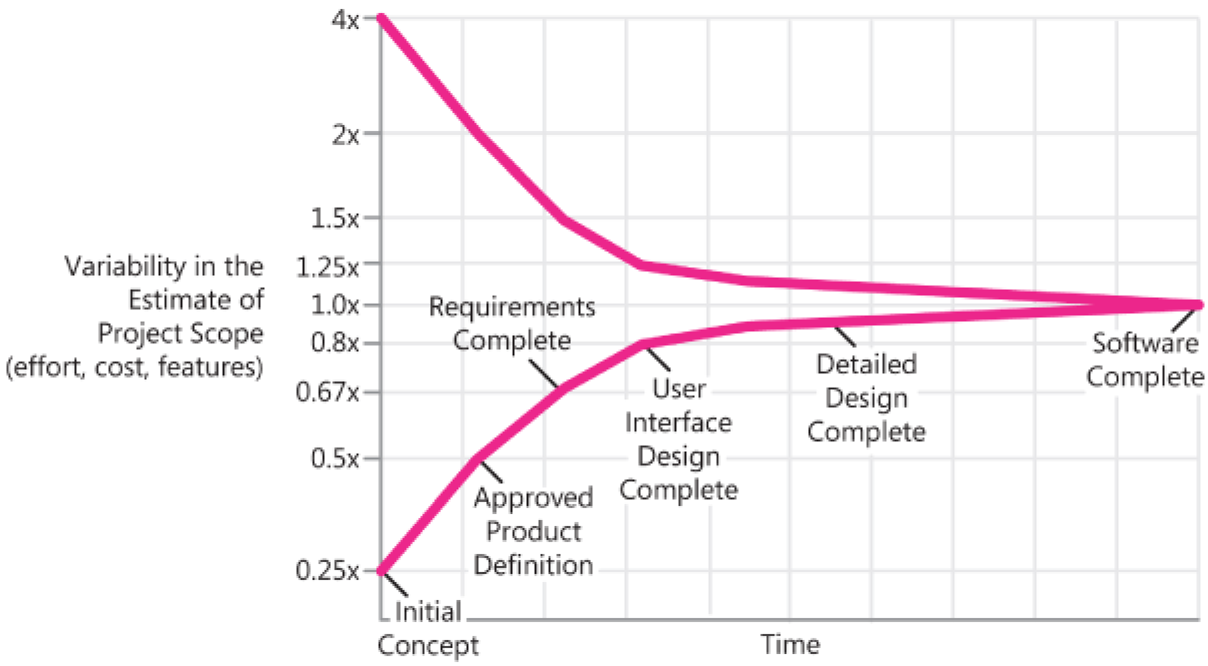


Figure 3 Cone of Estimate Uncertainty (from [5])

References

[1] CIS 895: MSE Project and Portfolio, Department of Computer and Information Sciences, Kansas Status University, http://mse.cis.ksu.edu/documents/MSE-portfolio.pdf (last accessed 10/24/2008).

[2] Barry Boehm, Bradford Clark, Ellis Horowitz, Ray Madachy, Richard Shelby, Chris Westland, "Cost Models for Future Software Life Cycle Processes: COCOMO 2.0," Annals of Software Engineering, (1995).

[3] COCOMO II Model Definition Model, Center for Software Engineering, University of Southern California, http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf (last accessed 12/28/2009).

[4] COCOMO II.2000.0 Software, Center for Software Engineering, University of Southern California.

[5] Software Estimation: Demystifying the Black Art, Steve McConnell, Microsoft Press, 2006.

[6] Project Plan: MSE Portfolio Project, Doug Smith, http://people.cis.ksu.edu/~dougs/Site/Phase1_Artifacts_files/project_plan.pdf

[7] Sonar code quality platform - http://www.sonarsource.org/