**Project Plan**

For Multiagent Control of Traffic Signals

Version 1.0

Submitted in partial fulfillment of the requirements of the degree of MSE

Bryan Nehl

CIS 895 – MSE Project

Kansas State University

Table of Contents

[1 Introduction 3](#_Toc311064037)

[1.1 References 3](#_Toc311064038)

[1.2 Terms 3](#_Toc311064039)

[2 Cost Estimate 4](#_Toc311064040)

[2.1 COCOMO 2.0 4](#_Toc311064041)

[2.1.1 Computing Unadjusted Function Points 4](#_Toc311064042)

[2.1.2 Estimate Variables 5](#_Toc311064043)

[2.1.3 Time to develop equation 6](#_Toc311064044)

[2.1.4 Early Design Calculations 7](#_Toc311064045)

[2.1.5 Discussion 8](#_Toc311064046)

[3 Architecture Elaboration Plan 8](#_Toc311064047)

[3.1 Revise Vision Document 8](#_Toc311064048)

[3.2 Revise Project Plan 8](#_Toc311064049)

[3.3 Create Formal Specification 8](#_Toc311064050)

[3.4 Create Architectural Design 8](#_Toc311064051)

[3.5 Create Test Plan 8](#_Toc311064052)

[3.6 Conduct Technical Inspection 8](#_Toc311064053)

[3.7 Create Executable Architecture Prototype 8](#_Toc311064054)

[4 Implementation Plan: Deliverables 8](#_Toc311064055)

[4.1 Action Items 8](#_Toc311064056)

[4.2 Technical Inspection Letters 8](#_Toc311064057)

[4.3 Component Design Document 8](#_Toc311064058)

[4.4 User Manual 8](#_Toc311064059)

[4.5 Source Code 8](#_Toc311064060)

[4.6 Assessment Evaluation 8](#_Toc311064061)

[4.7 Project Evaluation 8](#_Toc311064062)

[4.8 References 8](#_Toc311064063)

[5 Implementation Plan: Work Breakdown Structure 8](#_Toc311064064)

[5.1 Inception Phase 9](#_Toc311064065)

[5.2 Elaboration Phase 9](#_Toc311064066)

[5.3 Production Phase 9](#_Toc311064067)

[6 Maintenance? 9](#_Toc311064068)

# Introduction

This is the initial project plan for the Multiagent Control of Traffic Signals (MACTS) Masters of Software Engineering project.

## References

1. W. Royce, Software Project Management: A Unified Framework, Addison-Wesley,

1998, p. 34, pp. 265-281.

1. B. Boehm et al., “Cost Models for Future Software Processes: COCOMO 2.0,” Annals of

Software Eng., Vol. 1, 1995, pp. 57-94.

1. K-State Master of Software Engineering web site, “MSE Portfolio Requirements,”
2. November 28, 2011; http://mse.cis.ksu.edu/portfolio.html.
3. Center for Systems and Software Engineering web site, “COCOMO II,” December 4, 2011: http://sunset.usc.edu/csse/research/COCOMOII/cocomo\_main.html.
4. The Code Project web site, “Software Project Cost Estimates Using COCOMO II Model,” December 4, 2011: http://www.codeproject.com/KB/architecture/cocomo2.aspx.
5. Naval Postgraduate School web site, “COCOMO II - Constructive Cost Model,” December 4, 2011: http://diana.nps.edu/~madachy/tools/COCOMOII.php.
6. Center for Software Engineering, USC, COCOMO II: Model Definition Manual Version 2.1, 2000.
7. The Code Project web site, “Calculating Function Points,” December 4, 2011: http://www.codeproject.com/KB/architecture/Calculate\_Function\_Point.aspx.
8. USC Center for Software Engineering website, “COCOMO II Affiliates,” December 4, 2011: http://csse.usc.edu/csse/affiliate/private/COCOMOII\_Driver+Calc\_Ss/SpreadSheet-COCOMOII.html.

## Terms

COCOMO is short for COnstructive COst MOdel.

SLOC is an acronym for Source Lines Of Code. These are lines of code that are neither comment nor whitespace.

UFP is an acronym for Unadjusted Function Points.

**Data Functionality**

Internal Logical Files (ILF) are files that represent major logical groupings of systems data that are persisted.

External Interface Files (EIF) are files that are shared between software systems.

**Transaction Functionality**

External Inputs (EI) represent data that enters into the system.

External Outputs (EO) represent data that exits or is output by the system.

External Queries (EQ) are counted by examining portions of the system that accept an input and respond immediately with some form of output.

# Cost Estimate

## COCOMO 2.0

### Computing Unadjusted Function Points

Unadjusted function points are used in the early design stage for project estimation. I followed the information I found about computing function points on the Code Project website[9]. Additional information included definitions of the function point types can be found in section 2.2 of COCOMO II: Model Definition Manual[8]. I reviewed my use cases and arrived at the following table:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Use Case | ILF | EIF | EI | EO | EQ | Total |
| 1 | 2 |  |  | 1 |  |  |
| 2 |  |  |  | 2 | 2 |  |
| 3 | 1 |  |  | 1 |  |  |
| 4 | 1 |  |  | 1 |  |  |
| 5 |  | 1 | 1 | 1 | 1 |  |
| 6 | 1 |  |  |  |  |  |
| 7 | 1 | 1 |  | 1 | 1 |  |
| 8 | 1 |  |  | 1 |  |  |
| 9 |  |  | 1 | 1 |  |  |
| 10 | 1 |  |  | 1 |  |  |
| 11 | 1 | 1 |  | 1 | 1 |  |
| Sub Total | 9 | 3 | 2 | 11 | 5 |  |
| All weights Low | 7 | 5 | 3 | 4 | 3 |  |
| weighted | 63 | 15 | 6 | 44 | 15 | **128** |

### Estimate Variables

|  |  |  |  |
| --- | --- | --- | --- |
| Cost Driver | Value (Text) | Factor | Description |
| LANG | Hybrid 3rd/4th Python | 50 | Used for converting from UFP to SLOC |
| PREC | Nominal | 3.72 | Precedentedness |
| FLEX | High | 2.03 | Development Flexibility |
| RELY | Very High | 5 | Required Software Reliability |
| DATA | Low | 2 | Data Size |
| CPLX | Nominal | 4 | Product Complexity |
| RUSE | Low | 0.95 | Required reusability |
| DOCU | Nominal | 3 | Documentation to match life cycle needs |
| RESL | High | 1.41 | Architecture and risk resolution |
| TEAM | High | 2.19 | Team Cohesion |
| ACAP | High | 4 | Analyst Capability |
| PCAP | High | 4 | Programmer Capability |
| PCON | Very High | 5 | Personnel Capability |
| APEX | Nominal | 3 | Application Experience |
| PLEX | Nominal | 3 | Platform Experience |
| LTEX | Low | 2 | Language and Tool Experience |
| PMAT | High | 3.12 | Process Maturity |
| TIME | Very High | 5 | Execution Time |
| STOR | Nominal | 3 | Main Storage Constraint |
| PVOL | Low | 2 | Platform Volatility |
| TOOL | Nominal | 3 | Use of Software Tools |
| SITE | Extra High | 6 | Multisite Development |
| SCED | Nominal | 1.00 | Schedule |

### Time to develop equation

This model is explained in Section 4 of the COCOMO II Design Model Definition document [8].

(Equation 1)

**Symbol Description**

B The scaling base-exponent for the effort equation, currently set to 0.91

C Coefficient that can be calibrated currently set to 3.67

D Scaling base-exponent that can be calibrated currently set to 0.28

E The scaling exponent for the effort equation

PMNS Person-Months estimated without the SCED cost driver (Nominal Schedule)

SCED Required Schedule Compression

TDEV Time to Develop in calendar months

### Early Design Calculations

Early design calculations use composites of the earlier defined cost drivers. See Table B-7, p281 of Royce’s, Software Project Management[1] for a table on how to compute the process exponent.

|  |  |  |
| --- | --- | --- |
| RCPX | 1.33 | RELY + DATA + CPLX + DOCU  Product Reliability and Complexity |
| RUSE | 0.95 | RUSE  Developed for Reusability |
| PDIF | 1.29 | TIME + STOR + PVOL  Platform Difficulty |
| PREX | 1.12 | APEX + PLEX + LTEX  Personnel Experience |
| PERS | 0.63 | ACAP + PCAP + PCON  Personnel Capability |
| FCIL | 0.73 | TOOL + SITE  Facilities |
| SCED | 1.00 | SCED  Required Development Schedule |
| **EArch** | 0.8395 | Product of the above defined composite cost drivers. |
|  | | |
| **Size (KSLOC)** | 6.4 |  |
| Size (UFP) | 128 | Unadjusted Function Points |
| UFP->Lines of Code | 50 | Conversion factor from UFP to SLOC |
| PREC | 0.01 | Precedentedness |
| FLEX | 0.03 | Flexibility |
| RESL | 0.03 | Risk Resolution |
| TEAM | 0.04 | Team cohesiveness and communication |
| PMAT | 0.03 | Process Maturity |
| **Process Exponent** | 1.15 | (A sum of the PREC, FLEX, RESL, TEAM and PMAT parameters) |
|  | | |
| **Effort** | 17.39 | staff-months |
| **TDEV** | 2.82 | time to develop C=3.0, D=0.33, B=1.01 |
| **TDEV Early Design** | 3.47 | time to develop C=3.67, D=0.28, B=0.91 |
| **TDEV 1997 Calibration** | 2.50 | time to develop C=2.66, D=0.33, B=1.01 |

### Discussion

I computed the TDEV estimate with three different sets of constants. The results ranged from 2.5 to 3.47. This seems reasonable.

I had to be a little creative with the UFP to SLOC conversion because Python wasn’t listed in the documentation. I did some research and while Python is sort of 3rd generational it also incorporates more modern dynamic features. My experience is that Python code usually takes the same amount of SLOC or less to do the same thing as Java. Therefore, I choose a conversion rate of 50.

# Architecture Elaboration Plan

The following subsections detail the tasks that will be completed during the elaboration phase of the project.

## Revise Vision Document

## Revise Project Plan

## Create Formal Specification

## Create Architectural Design

## Create Test Plan

## Conduct Technical Inspection

## Create Executable Architecture Prototype

# Implementation Plan: Deliverables

Deliverables statement

## Action Items

adf

## Technical Inspection Letters

asdf

## Component Design Document

adsf

## User Manual

asfd

## Source Code

af

## Assessment Evaluation

asdf

## Project Evaluation

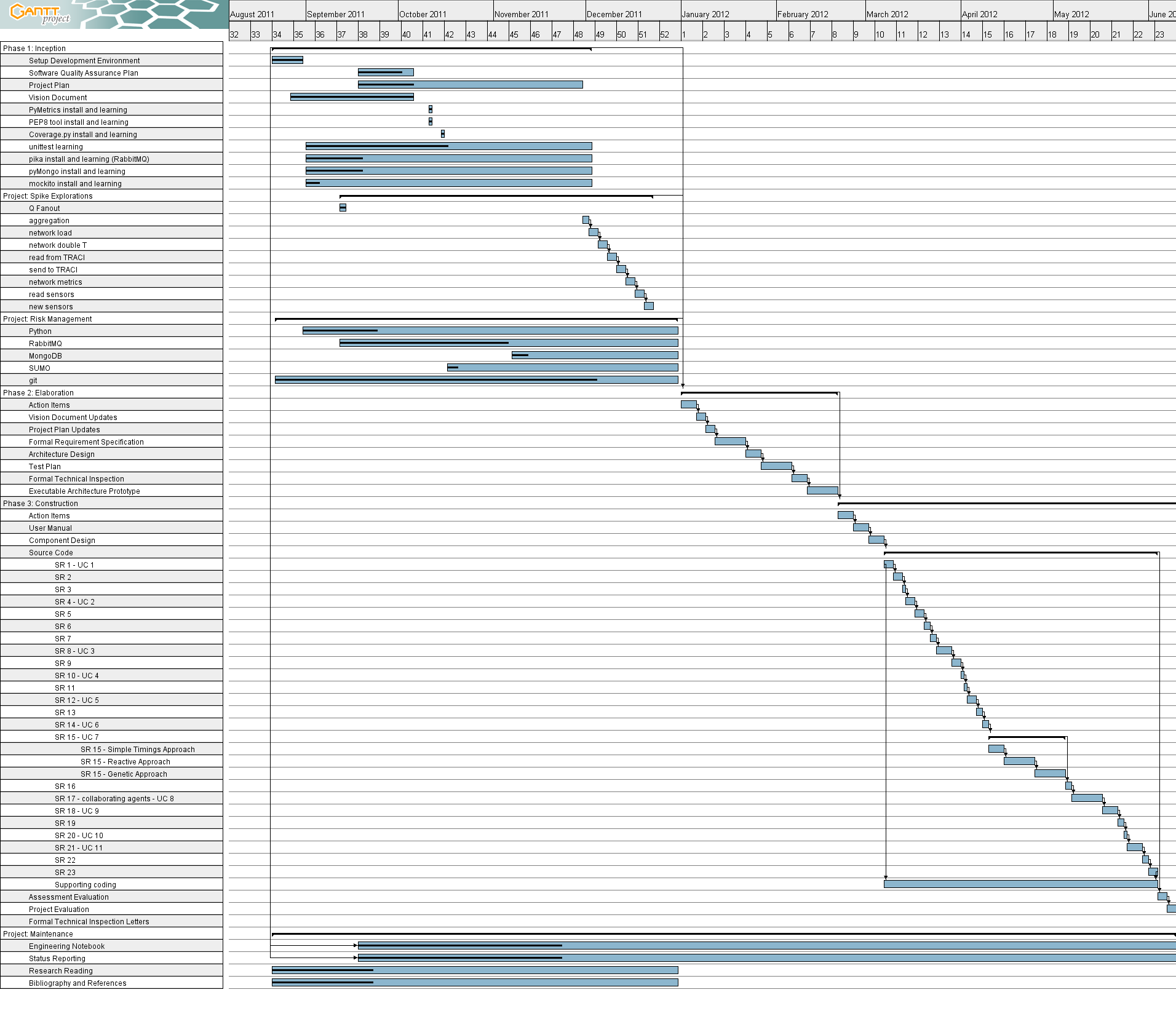
asdf

## References

adf

# Implementation Plan: Work Breakdown Structure

Words about the work breakdown structure here…



## Inception Phase

asdf

## Elaboration Phase

asdf

## Production Phase

asdf