2011

CSCI 4830-001
Intro to Software
Engineering
Spring Semester

Team Members
[redacted]

Prepared for:
Dr. Mansour Zand

[redacted]

SERBERUS

Serberus Planning Document

Revision/Change Record

Revision	Date	Revision/Change Description	Pages	User
0.0	2/28/11	Base Draft	All	SM
0.1	3/14/11	Section 5, Partial Section 7	9-14, 18-20	KH/DN
0.1	3/15/11	COCOMO Calculation Section 4, Section 6, Partial Section 7	8, 10, 14-16, 18	AM
0.1	3/16/11	Section 1-3, Partial Section 7	6-7, 18-19	SM
1.0	3/16/11	Editing and compilation of Final Document	All	SM

Table of Contents

1. Overview

1.1. Project summary

1.1.1. Purpose, scope, and objectives

The purpose of this document is to outline and provide context on how the Noloki development team will plan, design, and execute the implementation of a Serberus Fire and Security System.

1.1.2. Assumptions and constraints

Since this the product of this project, the Serberus security system, is the initial release from the company Noloki, the product will be built from the ground up, no reuse is available. The schedule for this project will be limited to the spring semester of CSCI 4830 section 001. Staffing for this project will remain at four people with no additional hires for the duration of the Serberus software development. The software will follow the requirements spelled out by the project definition (see annex i).

1.1.3. Project deliverables

The Serberus Fire and Security System will be designed to provide control over both the fire prevention system and security system for the building(s) where it is installed. Some of the features the system must accommodate are outlined in the Annex section at the end of this document.

1.1.4. Schedule and budget summary

The schedules of deliverables are designated as:

Section	Date
System Specification Document	01/31/11
Software Requirements Specification	02/21/11
Planning Document	03/19/11
Design Document	04/18/11
Alternate Analysis and Design Test plan/cases	04/18/11
Completed Project	04/25/11

The budget is outlined in full detail in section 5.2.4.

1.2. Evolution of the plan

The base draft of this planning document will reside with-in Noloki's dropbox (http://www.dropbox.com) account, which all members of Noloki will have access to. The official draft will contain a revision number and change record notifying all readers of updates and changes. All official and unofficial changes to this document will require updates to these sections. Previous versions of this document will also be stored in the dropbox for archiving.

2. References

- Software Engineering, 9th edition, Ian Sommerville, 2011
- Spring 2011 Term Project Requirements, Dr. Mansour Zand, Jan. 2011
- IEEE Std 1058, 1998 Ed, IEEE Standard for Software Project Management plans

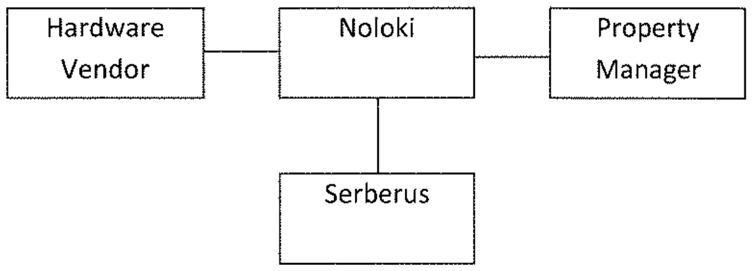
3. Definitions

- **COCOMO 2**: COnstructive COst MOdel 2is a model that allows one to estimate the cost, effort, and schedule when planning a new software development activity.
    - **Components of COCOMO**
        - *PM = Person-Months*
        - *A = Initial calibration*
        - *Size = Thousands of lines of code*
        - *B = Disproportionate effort for large projects*
        - *M = Product, Process, People attribute multiplier*
        - *PM_M = Amount of effort for auto generated code*
        - *PREC = Precedentedness*
        - *DFLEX = Development Flexibility*
        - *ARR = Architecture / Risk resolution*
        - *TCOH = Team Cohesion*
        - *PMAT = Process Maturity*
        - *PERS = Personnel Capability*
        - *RCPX = Product reliability and complexity*
        - *PDIF = Personnel Experience*
        - *PREX = Personal Experience*
        - *FCIL = Team support Facilities*
        - *SCED = Required schedule*
        - *ASOLC = Amount of effort for auto-generated code is based on the number of lines of code*
        - *AT = Percentage of automatically generated code in the whole system*
        - *ATPROD = Productivity level for this type of code*

4. Project organization

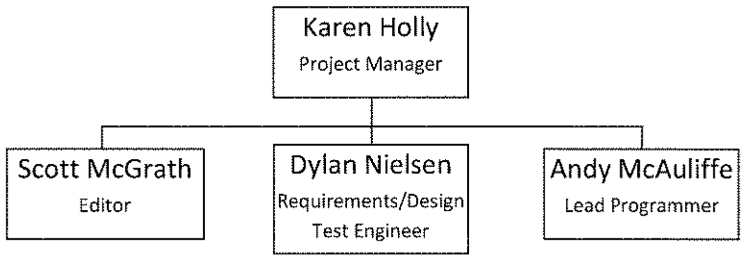
4.1. External interfaces

The Serberus Fire and Security System is a software product manufactured by the Noloki Corporation. Noloki works with the security system hardware vendor to interface the software with the security system on behalf of the manager of the property where the system will be installed.



4.2. Internal structure

The Noloki software engineering team consists of a project manager, an editor, a requirements/design test engineer, and a lead programmer, organized as per the following org chart.



4.3. Roles and responsibilities

Work Activity	Project Manager	Editor	Requirements /Design Test Engineer	Lead Programmer
overall project management	X			
document contribution	X	X	X	X
document editing		X		
document printing and		X		

distribution				
collaboration solution procurement			X	
requirements validation			X	
design validation			X	
configuration management				X
GUI programming	X			
interface/configuration design				X
component programming	X	X	X	X
unit testing	X	X	X	X
integration testing	X	X	X	X

5. Managerial process plans

The project plans of Serberus project include plans for: start, work, control, closeout, and risk management.

5.1. Start-up plan

This section details the estimation, staffing, resources and training for the Serberus Project.

5.1.1. Estimation plan

5.1.1.1 Cost

The cost of the project will be estimated by listing all expected expenses and totaling them. A 20% contingency will be added to the final estimate.

5.1.1.2. Schedule

The Serberus project schedule is shown below. Each date specifies the date on which the deliverable is due. Monday delivery dates may be moved to Wednesday dates on occasion. The absolute last date for delivery of the final is April 27, 2011.

Project Team Assembled	**January 17, 2011**
System Specification	January 31, 2011
Software Requirements Specification	February 23, 2011
Project Plan	March 18, 2011
Project Design	April 11, 2011
Alternate Analysis and Design	April 18, 2011
Test Plan, User Manual, Project Legacy and Compilation of Previous Documents	April 25, 2011

5.1.1.3 Resources

The Serberus Project shall require:

- Location for team meetings
- Computer and internet access for all developers
- Word editing software (Microsoft Office 2007/Open Office)
- The NetBeans IDE
- mySQL
- Access to http://www.dropbox.com

5.1.1.4 Basis of Estimation

Since this is a brand-new system, no reusable components are being utilized, and the requirements are well-documented, a COCOMO 2 -- Early Design model shall be employed to estimate the implementation effort required.

$PM = A \times Size^{B} \times M + PM_{M}$

$PM = person - months$
$A = initial\ calibration$
$\mathrm{Size} = thousands\ of\ lines\ of\ code$
$\mathrm{B} = disproportionate\ effort\ for\ large\ projects$
$\mathrm{M} = product,\ process,\ people\ attribute\ multiplier$
$PM_{m} = amount\ of\ effort\ for\ auto - generated\ code$

The remainder of this section builds the variables required to complete this equation.

<u>Initial Calibration</u>
$A = initial\ calibration = 2.94$

<u>Converting Function Points to Lines of Code for Java</u>
The following web sites declare 53 to be the number of lines of Java source code per function point. The handout from Dr. Zand does not include a Java conversion but does offer the number 29 for C++. C++ and Java are similar enough to make the 53 SLOC/UFP number plausible.

http://msquaredtechnologies.com/m2rsm/docs/reports/rsm_f_report.htm
http://www.qsm.com/?q=resources/function-point-languages-table/index.html

Serberus Screens

In the following list of screens for the Serberus system, the screen name is preceded by an 'S', indicating it is a simple screen for purposes of estimation, or 'A', indicating it is an average screen.

S Login Screen
S Task Screen
S Users Main Screen
S Add user screen
S Edit user screen
S Delete user screen
S Generic confirmation screen
S View log screen
S Configuration main screen
A Add zone screen
S Edit zone screen
A Edit Fire response screen
A Edit Security response screen
A Edit sensor screen
A Edit response screen
A Edit response protocol screen
A Edit response system screen
A Delete Zone Screen
S System Mode Screen
S Armed Mode main screen
S Test mode screen
A Zone test screen
A Auditory test screen
S Reset Mode screen
S Logoff Screen
S Change password screen
S View options screen
S View by zone screen
A View by sensor screen
A Display Screen

Estimating Number of Function Points
(# simple screens = 18) x (weighting factor = 3) = 54
(# average screens = 12) x (weighting factor = 5) = 60
(# simple external interfaces = 1) x (weighting factor = 5) = 5
(# average external interfaces = 1) x (weighting factor = 7) = 7
(# simple database table = 13) x (weighting factor = 7) = 91
Total Count = 54 + 60 + 5 + 7 + 91 = 217
Complexity Multiplier = 0.1
# Function Points = (Total Count = 225) x (Complexity = 0.1) = 21.7

Lines of Code Estimate
The estimate of the number of lines of code is based on the number of function points times the lines of code per function point factor.

ASLOC=21.7 Function Points X 53 ASLOC/Function Points = 1150
Size=*Thousands of lines of code*=1.150

Large Project Exponent
The disproportionate effort for large projects exponent is based on precedentedness, development flexibility, architecture/risk resolution, team cohesion, and process maturity. Each of these numbers ranges from 1 to 5 and reflect the level of the particular attribute, with 1 being "low measure" and 5 being "high measure".

$$B = 1.01 + \frac{PREC+DFLX+ARR+TCOH+PMAT}{100}$$

$PREC = precedentedness = 1$
$DFLX = development\ flexibility = 1$
$ARR = architecture/risk\ resolution = 2$
$TCOH = teamcohesion = 2$
$PMAT = process\ maturity = 1$

$$B = 1.01 + \frac{1+1+2+2+1}{100} = 1.01 + \frac{7}{100} = 1.08$$

Product Process People Attribute Multiplier
The product, process, people attribute multiplier is the product of personnel capability, product reliability and complexity, reuse required, platform difficulty, personnel experience, team support facilities, and the required schedule. Each of

these number ranges from 1 and 10, with 1 being "low measure" and 10 being "high measure".

$M = PERS \times RCPX \times RUSE \times PDIF \times PREX \times FCIL \times SCED$
$PERS = personnel\ capability = 2$
$RCPX = product\ reliability$ and $complexity = 1$
$RUSE = reuse\ required = 1$
$PDIF = platform difficulty = 2$
$PREX = personnel\ experience = 1$
$FCIL = team support\ facilities = 2$
$SCED = required\ schedule = 1$
$M = 2 \times 1 \times 1 \times 2 \times 1 \times 2 \times 1 = 8$

Amount of Effort for Auto-Generated Code

The amount of effort for auto-generated code is based on the number of lines of code (ASOLC), the percentage of automatically generated code in the whole system (AT), and the productivity level for this type of code (ATPROD).

$PM_M = \frac{ASOLC\ x\ (\frac{AT}{100})}{ATPROD}$

ASLOC = 22 Function Points x 53 ASLOC / Function Point = 1166
ATPROD = productivity level for this type of line of code = 1400
AT = percentage of auto generated code for whole system = 20

$PM_M = \frac{ASOLC\ x\ (\frac{AT}{100})}{ATPROD} = \frac{1166\ x\ (\frac{20}{100})}{1400} = \frac{233.2}{1400} = 0.17$

COCOMO 2 Early Design Estimate

$PM = A\ x\ Size^B\ x\ M + PM_M$
PM = 2.94 x $1.150^{1.08}$ x 8 + .17 = 28
The project is estimated to take 28 person-months to code and test.

5.1.1.5 Re-estimation

The required project resources and project deadlines will be periodically assessed at the weekly team meetings.

5.1.2. Staffing plan

Staffing is provided up to four people. These people will ideally work well together and have acceptable programming and project experience.

5.1.3. Resource acquisition plan

Resources shall be obtained as necessary and as decided by the team. We do not anticipate needing any expensive or difficult to install resource.

5.1.4. Project staff training plan

All staff of the Serberus project team will need to become familiar with NetBeans, Java, SVN and mySQL if they do not have prior experience with these tools.

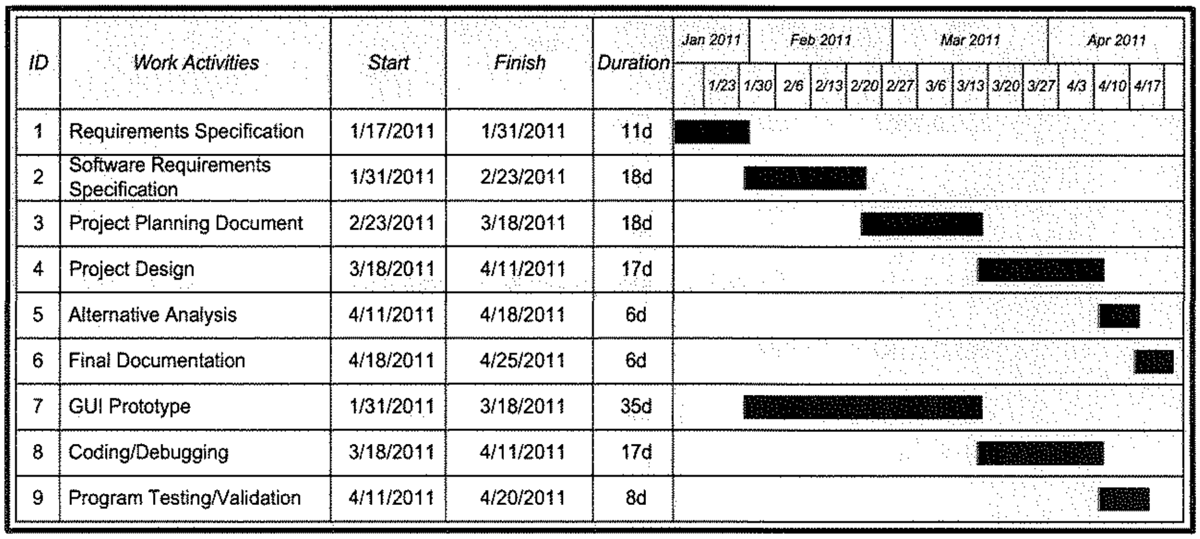
Staff will be trained prior to using a tool in the project; this training is anticipated to occur at least a week before usage of the tool. Training will be provided by the following members for the following tools:

Tool	Trainer	Date	Work Item
NetBeans	Karen Holly	3/1	7.3
Subversion	Andrew McAuliffe	3/18	8.1

5.2. Work plan

The Serberus Project will involve the following resources, budget and work activities.

5.2.1. Work activities



ID	Work Activities	Start	Finish	Duration
1	Requirements Specification	1/17/2011	1/31/2011	11d
2	Software Requirements Specification	1/31/2011	2/23/2011	18d
3	Project Planning Document	2/23/2011	3/18/2011	18d
4	Project Design	3/18/2011	4/11/2011	17d
5	Alternative Analysis	4/11/2011	4/18/2011	6d
6	Final Documentation	4/18/2011	4/25/2011	6d
7	GUI Prototype	1/31/2011	3/18/2011	35d
8	Coding/Debugging	3/18/2011	4/11/2011	17d
9	Program Testing/Validation	4/11/2011	4/20/2011	8d

1. Major work activities

ID	Work Items of Requirements Specification	Start	Finish	Duration	Jan 2011											
					17	18	19	20	21	22	23	24	25	26	27	28
1	Requirements Discovery	1/17/2011	1/21/2011	5d	■	■	■	■	■							
2	Requirements Validation	1/21/2011	1/25/2011	3d					■	■	■	■	■			
3	Drafting of User Requirements	1/25/2011	1/28/2011	4d									■	■	■	■
4	Finalization of Requirements Documents	1/28/2011	1/31/2011	2d												■

2: Breakdown of Activity 1

ID	Work Items of Software Requirements Specification	Start	Finish	Duration	Feb 2011																						
					1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	Use Cases	2/1/2011	2/8/2011	6d	■	■	■	■	■	■	■	■															
2	Discover System Objects	2/8/2011	2/15/2011	6d								■	■	■	■	■	■	■	■								
3	Drafting of Software Requirements	2/10/2011	2/18/2011	7d										■	■	■	■	■	■	■	■	■					
4	Finalize Software Requirements Doc	2/21/2011	2/23/2011	3d																					■	■	■

3: Breakdown of Activity 2

ID	Work Items of Planning Document	Start	Finish	Duration		Mar 2011																	
					28	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	Project Estimation	2/28/2011	3/4/2011	5d	■	■	■	■	■														
2	Drafting of Planning Document	3/4/2011	3/11/2011	6d					■	■	■	■	■	■	■	■							
3	Finalizing of Planning Document	3/11/2011	3/18/2011	6d												■	■	■	■	■	■	■	■

4 Breakdown of Activity 3

ID	Work Items of Project Design	Start	Finish	Duration	Mar 2011														Apr 2011										
					18	19	20	21	22	23	24	25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11
1	Documentation of Methods/Classes	3/18/2011	3/28/2011	7d	■	■	■	■	■	■	■	■	■	■	■														
2	Documentation of Databases	3/21/2011	3/24/2011	4d				■	■	■	■																		
3	Documentation of GUI	3/18/2011	3/24/2011	5d	■	■	■	■	■	■	■																		
4	Drafting of Design Document	3/24/2011	4/5/2011	9d							■	■	■	■	■	■	■	■	■	■	■	■	■						
5	Finalization of Design Document	4/5/2011	4/11/2011	5d																			■	■	■	■	■	■	■

5 Breakdown of Activity 4

ID	Work Items of Alternative Analysis	Start	Finish	Duration	Apr 2011							
					11	12	13	14	15	16	17	18
1	Method Analysis	4/11/2011	4/15/2011	5d	■	■	■	■	■			
2	Test Case Execution	4/11/2011	4/15/2011	5d	■	■	■	■	■			
3	Documentation	4/15/2011	4/18/2011	2d					■	■	■	■

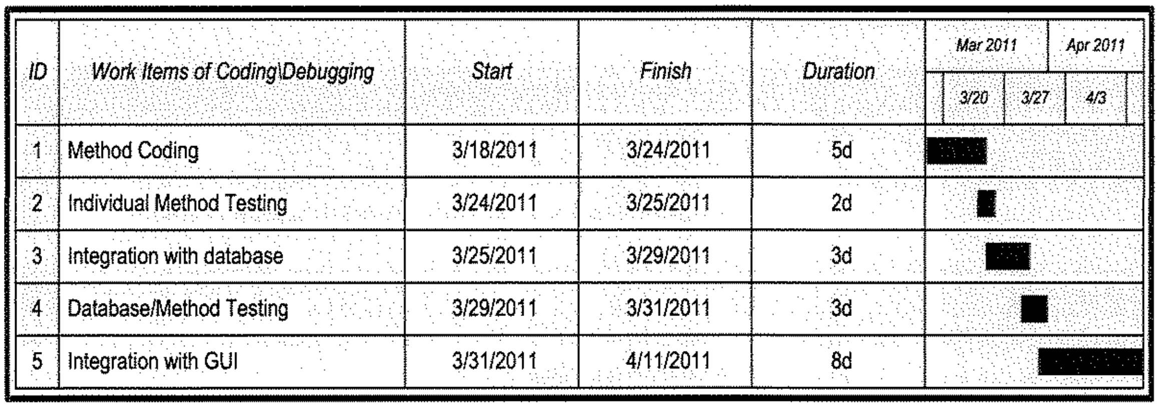
6 Breakdown of Activity 5

ID	Work Items of Final Documention	Start	Finish	Duration	Apr 2011									
					18	19	20	21	22	23	24	25	26	27
1	User Manual	4/18/2011	4/22/2011	5d	■	■	■	■	■					
2	Testing Documentation	4/18/2011	4/22/2011	5d	■	■	■	■	■					
3	Final Document Compilation	4/22/2011	4/27/2011	4d					■	■	■	■	■	■

7 Breakdown of Activity 6

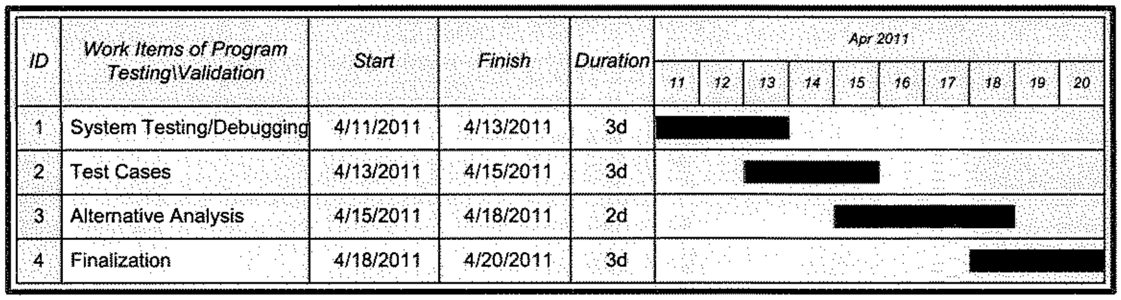
ID	Work Items of GUI Prototype	Start	Finish	Duration	Feb 2011				Mar 2011		
					1/30	2/6	2/13	2/20	2/27	3/6	3/13
1	GUI Design and Specification	1/31/2011	2/18/2011	15d	■	■	■				
2	GUI Validation	2/18/2011	2/28/2011	7d			■	■	■		
3	Finalize GUI elemnts	3/1/2011	3/18/2011	14d					■	■	■

8 Breakdown of Activity 7



ID	Work Items of Coding\Debugging	Start	Finish	Duration
1	Method Coding	3/18/2011	3/24/2011	5d
2	Individual Method Testing	3/24/2011	3/25/2011	2d
3	Integration with database	3/25/2011	3/29/2011	3d
4	Database/Method Testing	3/29/2011	3/31/2011	3d
5	Integration with GUI	3/31/2011	4/11/2011	8d

9 Breakdown of Activity 8

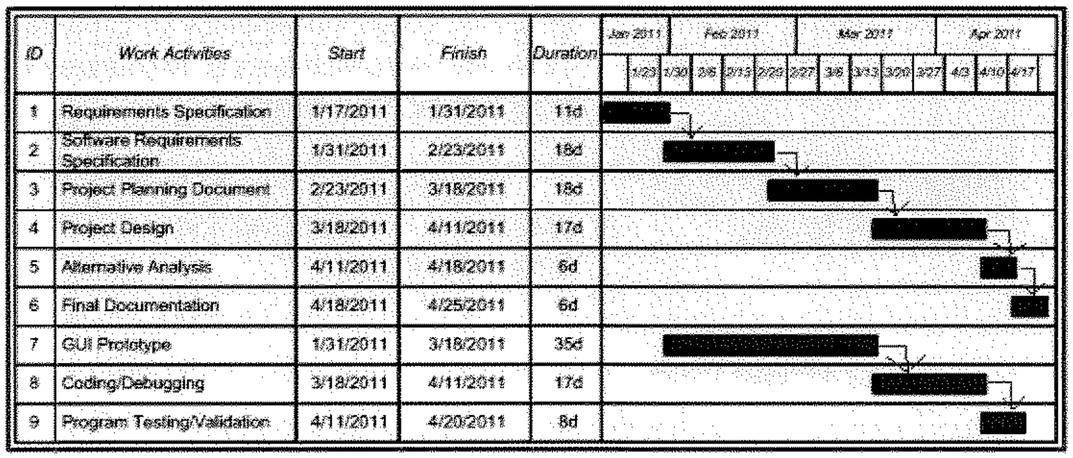


ID	Work Items of Program Testing\Validation	Start	Finish	Duration
1	System Testing/Debugging	4/11/2011	4/13/2011	3d
2	Test Cases	4/13/2011	4/15/2011	3d
3	Alternative Analysis	4/15/2011	4/18/2011	2d
4	Finalization	4/18/2011	4/20/2011	3d

10 Breakdown of Activity 9

5.2.2. Schedule allocation

The following diagram describes the nine main work activities and their relationships:



ID	Work Activities	Start	Finish	Duration
1	Requirements Specification	1/17/2011	1/31/2011	11d
2	Software Requirements Specification	1/31/2011	2/23/2011	18d
3	Project Planning Document	2/23/2011	3/18/2011	18d
4	Project Design	3/18/2011	4/11/2011	17d
5	Alternative Analysis	4/11/2011	4/18/2011	6d
6	Final Documentation	4/18/2011	4/25/2011	6d
7	GUI Prototype	1/31/2011	3/18/2011	35d
8	Coding/Debugging	3/18/2011	4/11/2011	17d
9	Program Testing/Validation	4/11/2011	4/20/2011	8d

Listed below are the milestones as determined for each major work activity. Each activity is ended with a deliverable.

5.2.2.1 Requirements Specifications

Milestone	**Acceptability Criteria**
Written List of Requirements	Agreement by all members of Noloki
Draft of System Requirements	Contains: System Definition, Diagram of System Architecture, Functional and Nonfunctional Requirements.
Deliverable	
Completed System Specification	Meets all criteria of draft and has been reviewed and approved by all team members.

5.2.2.2 Software Requirements Specifications

Milestone	**Acceptability Criteria**
Written List of Requirements	Agreement by all members of Noloki
Draft of Software Requirements Specification	Contains: Use cases, state diagrams and sequence diagrams for all major components.
Deliverable	
Completed Software Requirements Specification	Meets all criteria of draft and has been reviewed and approved by all team members.

5.2.2.3 Project Planning Document

Milestone	**Acceptability Criteria**
Determination of Schedule	Review of deadlines given by Dr. Zand, team member schedules
Draft of Planning document	Contains: Proposed Budget, Resources, Risk Management, and COCOMO 2 estimates
Deliverable	
Completed Planning Document	Meets all criteria of draft and has been reviewed and approved by all team members.

5.2.2.4 Project Design

Milestone	**Acceptability Criteria**
Written List of Classes/Methods	Agreement by all members of Noloki
Draft of Project Design	Contains: Detailed methods and classes

Deliverable	
Completed System Design	Meets all criteria of draft and has been reviewed and approved by all team members.

5.2.2.5 Alternative Analysis

Milestone	**Acceptability Criteria**
Completion of primary testing	All primary tests of the system have been completed, and any bugs solved or in progress of being solved
Draft of Analysis and Testing Document	Contains: List of all tests and outcomes
Deliverable	
Completed Alternative Analysis/Testing	Meets all criteria of draft and has been reviewed and approved by all team members.

5.2.2.6 Final Documentation

Milestone	**Acceptability Criteria**
Draft of User Manual	All major sections are included in user manual
Compilation of Previous Documents	All previous documents brought together. Any edits deemed necessary by the team made.
Draft of Project Legacy	Contains activity log, lessons learned by all members.
Deliverable	
Final Delivery of All Documents	Meets all criteria of draft and has been reviewed and approved by all team members.

5.2.2.7 GUI Prototype

Milestone	**Acceptability Criteria**
GUI layout	Diagrams detailing all main screens including logon, guard, and admin screens.
GUI Prototype	Small subset of main screens created and shown to team.
Deliverable	
GUI	All screens created in NetBeans. No functionality. Screens must be approved by all members of the team.

5.2.2.8 Coding/Debugging

Milestone	**Acceptability Criteria**
System Design	Diagrams detailing all major classes and associated methods
Major classes implemented	Implementation of major functionality of the system. Major methods should run without errors
Integration with database	Implementation of methods interacting with the database. Data should be added and edited in the databases correctly.
Integration with GUI	Main classes integrated as calls from the associated GUI screen. Methods and screen should be paired up, and GUI should contain major functionalities: configuring custom zones, allow the addition and editing of users, allow for system testing.
Deliverable	
Serberus System	Meets all previous criteria and has been reviewed and approved by all team members.

5.2.2.9 Program Testing/Validation

Milestone	**Acceptability Criteria**
List of Test Cases	Tests have been developed for the majority of the existing system
Completion of Test Cases	All designated test cases have been carried out on the system. Errors deemed major have been fixed. System works with minimal downtime and improper function
Integration Testing	System is made portable and moved to various systems for testing. Major systems tested will be: Mac OS, Windows 7,and Ubuntu Linux
Deliverable	
Serberus V 1.0	System fully functional and deemed operational by members of the team and demonstrated to customer.

5.2.3. Resource allocation

The following outline details the resources required for each work item.

5.2.3.1. Requirements Specification

1.1 All team members. Communication via email and team meetings.

1.2 All team members. Communication via email and team meetings.

1.3 All team members. Communication via email and team meetings. Word processing software.

1.4 All team members. Communication via email and team meetings. Word processing software.

5.2.3.2. Software Requirements Specification
2.1 One team member with word processing software.

2.2 One to Two team members

2.3 All team members and word processing software

2.4 One to Two team members with word processing software

5.2.3.3. Project Planning Document
3.1 One to two team members with word processing software

3.2 All team members and word processing software, diagram software

3.3 One to two team members with word processing software

5.2.3.4. Project Design
4.1 One to two team members, word processing software

4.2 One team member, word processing software

4.3 Two team members, word processing software, Netbeans IDE

4.4 All team members, word processing software

4.5 One to two team members, word processing software

5.2.3.5. Alternative Analysis/Testing
5.1 One to two team members, word processing software

5.2 All team members, Netbeans IDE

5.3 All team members, word processing software, Dropbox

5.2.3.6. Final Documentation
6.1 All team members, Netbeans IDE, word processing software

6.2 All team members, word processing software

6.3 All team members, word processing software, Dropbox

5.2.3.7. GUI Prototype
7.1 One to two team members, Netbeans IDE

7.2 All team members, Netbeans IDE

7.3 All team members

5.2.3.8. Coding/Debugging
8.1 All team members, Netbeans IDE, SVN

8.2 All team members, Netbeans, SVN

8.3 All team members, Netbeans, SVN, mySQL

8.4 All team members, Netbeans, SVN

8.5 All team members, Netbeans, SVN

5.2.3.9. Program Testing/Validation
9.1 All team members, Netbeans, SVN
9.2 All team members
9.3 All team members
9.4 All team members

5.2.4. Budget allocation
The anticipated budget is presented in United States Dollars (USD). By primarily using open source software, the budget is reduced considerably.

Item	Estimated Cost
Printed Documentation	per page: $0.07 X 170 Subtotal: $11.90
Printed Documentation Binding	Per document: $5 X 8 Subtotal: $40
Noloki Personnel Salary	$100,000/yr X .33 X 4 Subtotal: $132,000
Total:	$132,051.90
Contingency rate:	20%
GRAND TOTAL	$158,462.28

5.3. Control plan

The Serberus project will use the following measures to determine project status and control the project resources.

5.3.1. Requirements control plan

Requirements will be regularly assessed by the team at weekly meetings. The team members shall regularly review any previously written documentation. Any ambiguity will be discussed with the requestor, Dr. Zand.

5.3.2. Schedule control plan

The team's project manager, Karen, will see to it that all members of the team are aware of the major deadlines. These deadlines will be reviewed at all team meetings.

5.3.3. Budget control plan

Because budget is not expected to be a concern, the team will not focus heavily on the budget. Resources will be allocated to the Project delivery manager, Scott, as necessary. The team will review any additions to the budget at the weekly team meetings.

5.3.4. Quality control plan

The quality of work done will be assessed and reviewed by each member of the group. Coding will be verified by the Lead Programmer (Andrew McAuliffe), while other validations may pass onto Testing Engineer (Dylan Nielsen). If problems or errors are encountered, the person may be notified by any member of the team through email or at the team meeting.

5.3.5. Metrics collection plan

The metrics will be evaluated based on the timeline given by the Project Manager (Karen Holly). They will be collected by the Editor (Scott McGrath) at least two days before the projected due date via email or dropbox. The Editor will evaluate the veracity of the information and can decide whether the work needs to be redone. If coding is involved, the code will be verified by the Lead Programmer and possibly by the Testing Engineer. Reporting of the metrics will be given by the Dr. Zand. The group will then discuss development changes at the next meeting based on Dr. Zand's comments.

5.4.Risk management plan

5.4.1 Determination of Risk Factors

Risk factors in the area of technology, people, organization, tools, requirements, and estimation are analyzed and determined by the Test Engineer and placed in a table.

5.4.2 Prioritization of Risk Factors

Very High/Critical rates with high probability are dealt with first. Trickle down to the less harmful.

5.4.3 Contingency Planning Procedures

The Test Engineer is responsible for advising the group on the contingency plan. He or she may also talk to the project manager or inform the group at a meeting.

5.4.4 Updating Risk Factors

After every milestone, new potential risks and evaluation of old risks will occur by the Test Engineer.

5.4.5 Risks

Risk	Type	Probability	Effect
Staff turnover – Key people leave	Project	Low	Catastrophic
Requirements Change with Major design rework proposed	Project/ Product	Moderate	Serious
Software tools (Subversion/NetBeans) can't support project	Project/ Product	Low	Catastrophic
Time required to develop software is underestimated	Project/ Product	Moderate	Catastrophic
Loss or corruption of source code	Project/ Product	Low	Serious
Medical Leave/Family emergency for Staff	Project/ Product	Moderate	Serious
Competitor releases similar product to market first	Product	Low	Moderate
Patent infringement litigation	Product	Low	Moderate/ Serious
Corporate espionage	Product	Low	Serious
Building code changes (product installs)	Product	Low	Low
Loss of access to the internet (staffing communication)	Project	Low	Moderate
General Litigation	Project	Low	Moderate
Customer cancels project order	Project/ Product	Moderate	Catastrophic

5.5.Closeout plan

5.5.1 Documentation

At the end of the Serberus Project, all documentation associated with the project will be revised and compiled. A document containing the lesions learned by the staff will be written and added to the final compilation. Copies of all documentation will be distributed to the team, as well as to Dr. Zand.

5.5.2 Project Materials

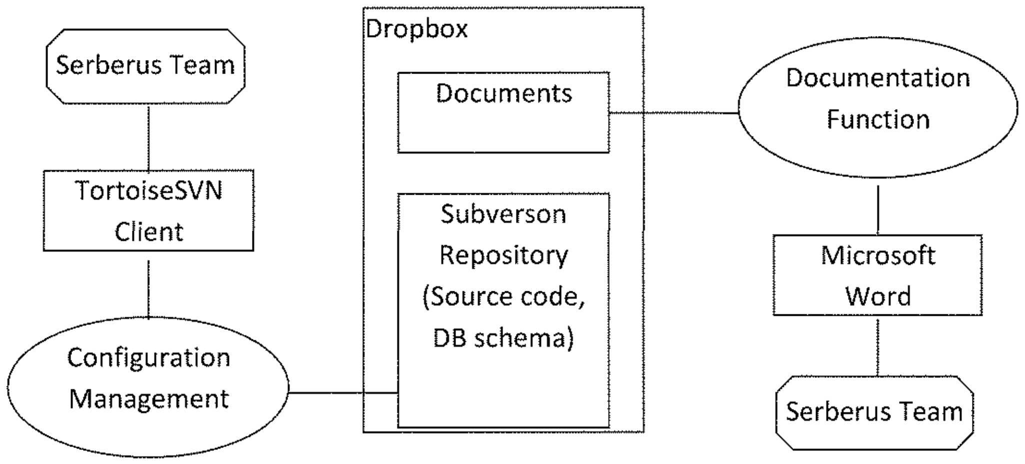
Copies of all programs pertaining to the Serberus project will be distributed to the team.

6. Technical process plans

6.1.Process model

Work Products Flow

The Serberus team shall produce all project documents using Microsoft Word. Work collaboration will be performed using the Dropbox file sharing folder. All source code and database schema code developed will be managed using the Subversion configuration management system. The Subversion repository will also be stored in a Dropbox folder for shared access.



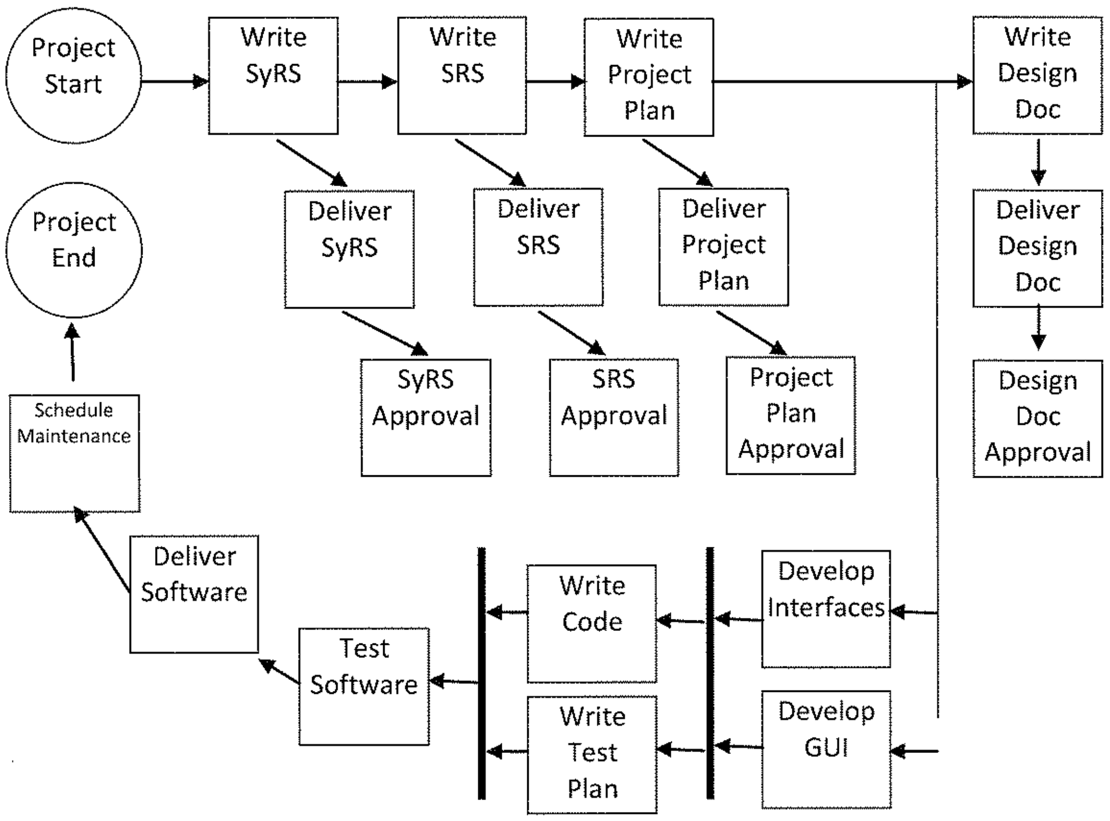
Work Products Delivery Timing

Deliverable	Delivery Date
System Specification Document	1/31/2011
Software Requirements Specification Document	2/21/2011
Planning Document	3/18/2011

Design Document	4/11/2011
Alternate Analysis and Design Document	4/18/2011
Test Plan and Test Cases Document	4/25/2011
Functional and Tested Software Product	4/25/2011

Dr. Zand will review and approve each work product delivered during the week after delivery.

Project Process Model



6.2.Methods tools and techniques

Methodology

A modified waterfall methodology shall be employed for the development of the documents and software of the Serberus system. The Noloki team shall obtain approval of the system requirements specification and software requirements specification before constructing the design document and proceeding with software development. Work on the design document and the software will commence concurrently, with incremental milestone deliverables submitted for approval.

Programming Languages Used

The GUI and back-end processing subsystems of Serberus shall be developed using the Java language. The MySQL dialect of the SQL language shall be used for all database interaction.

Tools Used

The following work activities shall be accomplished using the corresponding tool(s).

Work Activity	**Tool Used**
project management	E-mail
document preparation	Microsoft Word
software development	NetBeans IDE
database schema development/management	MySQL
source code configuration management	Subversion/Dropbox
document collaboration	Dropbox

6.3.Infrastructure plan

Computers belonging to each member of the Noloki team will be used to develop the documentation and code for the Serberus project. The software will be developed on a Microsoft Windows operating system, version XP or greater.

Each team member will have the following software installed on their computer: Microsoft Word (or equivalent), NetBeans IDE, TortoiseSVN, MySQL.

6.4.Product acceptance plan

Dr. Zand shall represent the acquiring organization.

All documentation pertaining to the project, including but not limited to the SyRS, SRS, Project Plan, Software Design, and Validation Plan, shall be reviewed and evaluated by Dr. Zand, in accordance with standard measures for acceptability for the CSCI 4830 course.

The software shall be submitted to Dr. Zand in both hard copy form (source code only) and binary form (source code and executable code). Dr. Zand shall determine the acceptability of the code by using a test client to simulate various security and fire detection scenarios. The following methods and criteria shall be used to determine the acceptability of the specified evaluation unit.

Evaluation Unit	Method Used	Acceptability Criteria
Serberus correctly alerts in the event of a confirmed fire.	demonstration	Configured actions for fire response are performed.
Serberus alerts and takes correct action in the event of a confirmed security breach, in accordance with current system configuration.	demonstration	Configured actions for security breach response are performed.
Serberus creates correct log entries based on detected events.	inspection	Log entries matching simulated events are created in the system log file.
Serberus produces reports based on log entries.	demonstration	A report is produced that is consistent with the system log.

7. Supporting process plans

7.1. Configuration management plan

Project documents shall be saved in a working directory in Dropbox -- a shared file collaboration tool that enables all team members to seamlessly synchronize folder contents with each other.

Working copies of project documents shall be saved in an "in progress" folder in Dropbox. Files containing contributions from individual team members shall be named with the member's name or initials as a component of the file name.

Final copies of project documents shall be saved in a "final documents" folder in Dropbox. These are the versions of the documents that are submitted for evaluation and acceptance.

Subversion version control software (http://subversion.apache.org/) shall be used as a configuration management solution for source code and configuration files for the project. The Subversion repository shall be established in a sub-folder of the Dropbox folder, where it shall be accessible by all team members.

Team members shall create working copies of the repository source code for purposes of creating new files and changing existing files. Changes shall be unit tested thoroughly prior to being committed to the repository. All changes, including newly created files, shall be committed to the repository, along with comments indicating the business reason(s) for the change(s).

It is a policy of Noloki that at all times, all code committed to the Subversion repository shall have a minimum level of quality. That minimum standard is that all such code shall not cause compilation errors.

7.2.Verification and validation plan

Based on the milestones set by the Project Manager (Karen Holly), the project will be subject to prototyping, simulation and modeling. First, the GUI prototype will verified and validated, then the coding will be next. The degree of independence between these development activities is _____. The GUI will be validated through the use of NetBeans and the project will be validated through Subversion. Testing, demonstration, and analysis will also be used with these tools.

7.3.Documentation plan

Documentation	Deliverable: Y/N	Date
System Specification Document	Y	01/31/11
Software Requirements Specification	Y	02/21/11
Planning Document	Y	03/19/11
Design Document	Y	04/18/11
Alternate Analysis and Design Test plan/cases	N	04/18/11
User Manual	Y	04/25/11
Project Planning Map	N	01/31/11
Budget Report	N	02/21/11
Source Code	N	04/25/11
Online Help Manual	N	04/25/11

7.4.Quality assurance plan

7.4.1 Communication during development

The customer shall be consulted regularly throughout the development process to ensure that all requirements are being adhered to. The customer shall be shown demonstrations of the components of the system as necessary.

Developers shall consult the Software Requirements and Systems Specification documents throughout the project lifespan.

7.4.2 Installation

The Serberus system shall be installed and initially configured by qualified personnel. The system shall pass initial tests before the system is declared operable.

7.4.3 Support

The Serberus system will be regularly inspected by Noloki personnel.

7.5. Reviews and audits

The project will be subject to review on a weekly basis to track progress and ensure deadlines are met. Customer is allowed to audit the process by request as long as it is submitted 48 hours in advance.

7.6. Problem resolution plan

7.6.1 Problem reporting

Any problem encountered by any member of the team shall be reported via email to the entire team.

7.6.2 Problem analyzing and prioritizing

Any problems not able to be resolved by the team via email or within 24 hours of the problem being reported to the team shall be examined by the following criteria to determine whether the problem is critical or not:

Evaluation Question	Yes	No
Does problem keep other members from working on their work item?	Critical	Not critical
Does the problem interfere with the completion of a large milestone?	Critical	Not critical
Will fixing the problem put the team in jeopardy of missing a deadline?	Not critical if answer to previous two is no.	Critical only if one of the previous answers was yes.

If the issue is not critical, but a solution is not found within 24 hours, the problem may be discussed at the next weekly team meeting, or an additional team meeting may be called.

7.7. Subcontractor management plan

No subcontractors will be used for the development of the Serberus Security and Fire Prevention System. All work will be performed by Noloki team members.

7.8. Process improvement plans

After consultation with Dr. Zand, the project may be subjected to re-assessment. Based on information by Dr. Zand, the group will discuss if the project should be implemented differently or documentation should be changed. If the decision is to modify, tasks will be divided based on role (Lead Programmer, Editor, Project Manager, or Test Engineer) and a new deadline will be set.

8. Customer Confidentiality

The nature and configuration of the Serberus system as installed and maintained by the customer shall be kept strictly confidential.

Annexes i

A fire and security alarm monitoring system

A large building may require an automated alarm system which monitors and controls all fire and security alarms in the building. Normally, the building is divided into zones and a number of alarms are associated with each zone. Alarms alert a central manned control area who may pass these on to the emergency services or may respond personally.

Factors which have to be taken into account in building such a system are:

- If the control area is unmanned and an alarm is activated this alarm should not be ignored if it is potentially serious. Emergency services should be automatically called.
- Some but not all parts of the building may be equipped with sprinkler systems or systems to shut down electrical equipment. These should be activated if a fire alarm is confirmed. They should not be activated if there are people in the same room.
- The building may be equipped with direction indicators which illuminate the route to the nearest exit. These should be activated when a fire alarm is confirmed. At the same time, an audible signal should sound alerting occupiers to leave the building.
- A security alarm may cause some internal doors to be locked automatically. It should be possible to isolate complete zones by automatic door locking.
- False alarms are common and it might be normal practice to have an alarm confirmed before alerting emergency services. There are different ways of confirming an alarm. In the case of a fire alarm, it may be confirmed by multiple sensors detecting a problem.

Develop a system to control such a monitoring system. The control panel is equipped with 2 monitors that one of them is used for activation and operation commands. The other monitor is used to display information about sensors, sprinkles, locking devices, alarms. Screens should be able to display several types of information at the same time.

The command center monitor should be very user friendly, easy to work with, and fast to operate. Simulation of sensors and other devices should include activating of several of them at the same time. In order to make implementation straight forward use limited number of sensors (20) and 5 of each other devices.