**THE GOATS - CIS 375 (SPMP) - Fall 2019**

**1.0 Introduction**

This section provides an overview of the software engineering project.

**1.1 Problem statement**

Emacs (a software company) currently has a less optimal method of collecting information from their departments in order to prove that they are following the internal published procedures, regulations, and laws set by outside and internal entities. Since people are inconsistent it’s tough to do everything manually expecting to get consistent and reliable results. However, automating the audit process, and eliminating as much human errors as possible, as well as tracking the progress of the audit, will be a viable solution.

**1.2 Project scope**

**1.2.1 Inclusions**

**1.2.1.1 Inputs**

* Updates for STENERS
* STENER for questioning
* Submitted STENERS
* Data for backing up answers for STENERS
* Due date for STENER

**1.2.1.2 Processing Functionality**

* Retrieve up-to-date STENER and overwrite current STENERS
* Send answers for STENER with corresponding data to oversight team.
* Forms to input answers/data.
* Refusing a user who does not have the credentials for a STENER having access
* STENER progress filtering
* Log out users after the idle time
* Add/Remove departments

**1.2.1.3 Outputs**

* STENER report (submitted STENER that have been sent back to oversight team)
* Completion Percentages or values of User STENER on Oversight team review page
* Sends email for report but needs signing in to enter report.

A description of the software is presented. Major inputs, processing functionality and outputs are described without regard to implementation detail.

**1.2.2 Exclusions**

List any exclusions from the project

* The printing of the documents submitted to the STENER report.
* The read/write/saving of a STENER outside of the application and the application's restrictions.
* Log out users after the idle time
* Add/Remove departments
* Sends email for report but needs signing in to enter report.
* Retrieve up-to-date STENER and overwrite current STENERS
* Data for backing up answers for STENERS

**1.3 Major software functions**

A functional decomposition of the software (for use during estimation and scheduling) is developed here. May be outline/bullet points or decomposition diagram format.

EMACS SOFTWARE STARTUP

* LOGIN ------ EQ(L) EIF(L)
* Oversight
  + Receive a submitted STENER ---- EI(L)
  + Create a new STENER ---- EI(L) ILF(L)
  + Adding/Editing/removing questions on STENER ---- EI(H) EO(H)
  + Adding/removing new departments
  + Approving/rejecting STENER ---- EI(L)
  + Report Page
  + Can make Violation logs when rejecting STENER ---- EI(A)
* Department
  + Answer current STENER ---- EI(L)
  + Submit STENER to oversight team. ---- EI(L)
  + Submit documents to questions in STENER ---- EI(A)
  + Able to view violation logs associated to their department ---- EO(A) EQ(L)
* STENER
  + Due date for STENER to be submitted, STENER is marked late. ---- EI(L) ILF(L)
  + Idle timer that saves current document and logs user out
  + Save current document when anything is submitted to it. ---- EI(A)
  + Checks if STENER is updated by oversight
  + Sending of email to departments.

**1.4 Performance/Behavior constraints**

Any special requirements for performance or behavior are noted here. Non-functional constraints such as on-line response time or batch window timeframe to complete processing.

* Runs on a server, as to prevent saving of STENERS from an outside source.

**1.5 Management and technical constraints**

**1.5.1 Management constraints**

Any special constraints that affect the manner in which the project will be conducted (e.g., limited resources or 'drop dead' delivery date) are noted here.

* Due Date: Dec 17 - 2019
* Not allowed to add any new members to the team.

**1.5.2 Technical constraints**

Any special constraints that affect the technical approach to development are noted here.

* Mastery of SQL/C# server and GUI

**2.0 Project Estimates**

This section provides cost, effort and time estimates for the projects

**2.1 Historical data used for estimates**

|  |  |
| --- | --- |
| Name | Past Experience |
| Muaz Alhaidar | C++ experience with OOP programming and some intermediate experience with search algorithms |
| Zakariya Ahmed | Intermediate Linux  Experience in web scripting  Bash/email scripting  C/C++ indeterminate experience  Doom map experience |
| Mostafa El Majzoub | C++ experience, java experience (a bit rusty), entry level experience in Linux. Some game modding experience. |
| Kevin Longworth | Mixed length projects and labs  In previous CIS classes |
| Cameron Labut | C# experience: (attempting to write a POS system…) writing installers for video game mods, writing bots to automate repetitive/tedious tasks in various video games... |

Describes the historical data that is relevant to the estimates presented. This is, what historical work can be used as the basis for your estimations (e.g. prior classwork projects, work projects)

**2.2 Initial Estimate**

Based solely on 1.0 and 2.1 details provide.

**2.2.1 SPMP Completion Estimate**

Estimated effort in hours to complete SPMP

12 hours

**2.2.2 Overall project estimate**

**2.2.2.1 Line-of Code Estimate**

What is the estimated lines of code to complete project

1350 lines

What is estimate in hours to complete project

220 hours

**2.2.2.2 Function Estimate**

What is the count of the functionality listed in 1.3

|  |  |  |  |
| --- | --- | --- | --- |
| Components | Complexity: |  |  |
|  | Low | Avg | High |
| EI | 4 | 5 |  |
| EO | 2 |  | 1 |
| EQ | 1 | 1 |  |
| ILF | 1 | 1 |  |
| EIF | 1 | 1 | 1 |

(4\*3)+(5\*4)+8+14+(7+10)+(5+7+10) = 93 Function Points

36 + 12 + 6 + 14 + 5 = 73

What is estimate in hours to complete project

744

73 \* 8 = 584

**2.2.2.3 Tasks Estimate**

What are high level tasks to complete (this is not a project plan) but tasks to think about for this estimate

* Premintatory Documentation
* Requirements
* Analysis
* Design
* Building
* Testing
* Deploying

What is estimate in hours to complete project

Before: 900 hours

Now: 430 hours

**2.2.2.4 Total overall project time estimate in hours of effort**

Team consensus on overall effort based on estimated values in 2.2.2.1 – 2.2.2.3 plus 2.2.1 and project management effort

Include SPMP Creation and Total Project

Before: (660 + 900 + 744 ) / 3 + 12 = 780 hours

Now: (220 + 430 +584 )/3 + 12 = 423 hours

Final estimate: 423hours

**2.3 Estimation techniques applied and results**

A description of each estimation technique and the resultant estimates are presented here. **DO NOT COMPLETE THIS SECTION UNTIL REMAINDER OF SPMP IS COMPLETED AND REVIEWED!**

**2.3.1 Estimation technique 1 – lines of code**

Tables or equations associated with lines of code estimation technique m are presented. Re-estimate lines of code based on completed SPMP.

|  |  |
| --- | --- |
| Name of Function | LOC |
| Oversight Functions | 500 (400) |
| Department Functions | 200 |
| STENER | 400 (200) |
| GUI | 150 |
| Server | 350 (200) |

Before: 500 + 400 + 200 + 150 + 350 = 1600 Lines of code

Now: 400 + 200 + +200 +150 + 200 = 1150 Lines

**2.3.2 Estimate for technique 1 – lines of code**

Estimate in hours generated for lines of code technique.

Before: 1600 LINES \* 30 MIN PER LINE = 800 Hours

Now: 1150 LINES \*15 MIN per line = 288 Hours

**2.3.3 Estimation technique 2 – function points**

Tables or equations associated with lines of code estimation technique m are presented. Re-estimate functionality and apply function point counting technique based on completed SPMP.

|  |  |  |  |
| --- | --- | --- | --- |
| Components | Complexity: |  |  |
|  | Low | Avg | High |
| EI | 4 | 5 | 0 |
| EO | 2 | 0 | 1 |
| EQ | 1 | 1 | 0 |
| ILF | 1 | 1 | 0 |
| EIF | 1 | 1 | 1 |

Before: (4\*3)+(5\*4)+8+14+(7+10)+(5+7+10) = 93 Function Points

Now: 36 + 12 + 6 + 14 + 5 = 73

**2.3.4 Estimate for technique 2 – function points**

Estimate in hours generated for function point technique.

Before: 8 \* 93 = ~750 hours

Now: 73 \* 8 = 584 hours

**2.3.5 Estimation technique 3 – process/task**

|  |  |
| --- | --- |
| Task | Hours |
| Premintatory Documentation | 20 (15) |
| Requirements | 10 |
| Analysis & Design | 250 (100) |
| Building | 250 (200) |
| Testing | 250 (13) |
| Deploying | 50 (3) |

**2.3.6 Estimate for technique 3 – process/task**

Before: 20+10+250+500+250+100=830

Now: 15 +10 + 100 +200 + 13 + 3 = 341

**2.4 Reconciled Estimate**

The final cost, effort, time (duration) estimate for the project (at this point in time) is presented here. Explain your rationale for these numbers compared to the three techniques above.

Before: Time: ~800 hours

Now: 450 hours

Cost: 0.00$

Effort: ~1600 lines of code

Now: ~1350 lines

**2.5 Project Resources**

People, hardware, software, tools, and other resources required to build the software are noted here.

* The Dearborn Goats staff are the main developers
  + Zakariya
  + Muaz
  + Kevin
  + Cameron
  + Mostafa
* Visual Studio is the standard IDE
* C#/SQL for building the backend
* C# for building the frontend
* Server for STENER synchronous
* Github for version control
* Microsoft Visio for the ERD
* Gitkraken: Git GUI client

**3.0 Risk Management**

This section discusses project risks and the approach to managing them.

**3.1 Project Risk Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Probability | Impact | Mitigation Plan | Contingency Plan |
| Failure to complete tasks | High | The critical path will lag behind, and the final project will be submitted in an uncompleted state  MED | Weekly goals are set, as to identify errors ahead of time. | Descope |
| Developer withdrawal from course | Low | More work is divided on remaining developers  HIGH | Lift stress, by dividing work more evenly. | Work is divided depending on expertise in the role that was vacated. Documentation about what they were doing before they left. |
| Lack of communication | Medium | Lack of needed communication to developers  MED | Thorough documentation of code and what they were working on | Keep track of teammates through communication clients. |
| Data Loss | Low | Attempting to redo progress, need to backtrack, resolving problems  CRITICAL | Data backups | Look at backups/Git Logs |

**3.2 Overview of Risk Mitigation, Monitoring, Management**

How will Risk Mitigation, Monitoring and Management be handled. Communication plan, when to apply the mitigation strategy, etc.

* Weekly goals are set, as to identify errors ahead of time. (REVIEW WITH GROUP)
* When to apply mitigation plan:
  + Assigning a value to the risk
  + When the value is higher than a certain threshold mitigation plan goes into effect.

**4.0 Project Schedule**

This section presents an overview of project tasks and the output of a project scheduling tool.

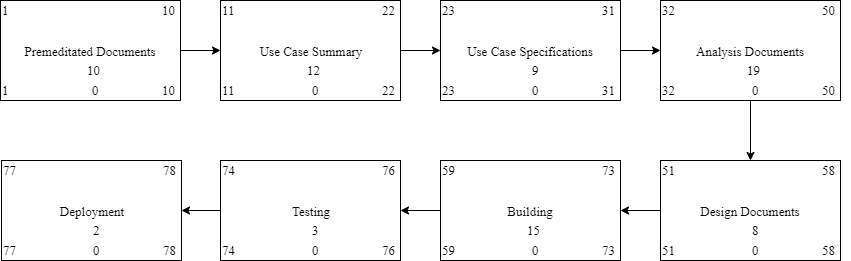
**4.1 Project task set**

* Premintatory Documentation
* Requirements
* Analysis
* Design
* Building
* Testing
* Deploying

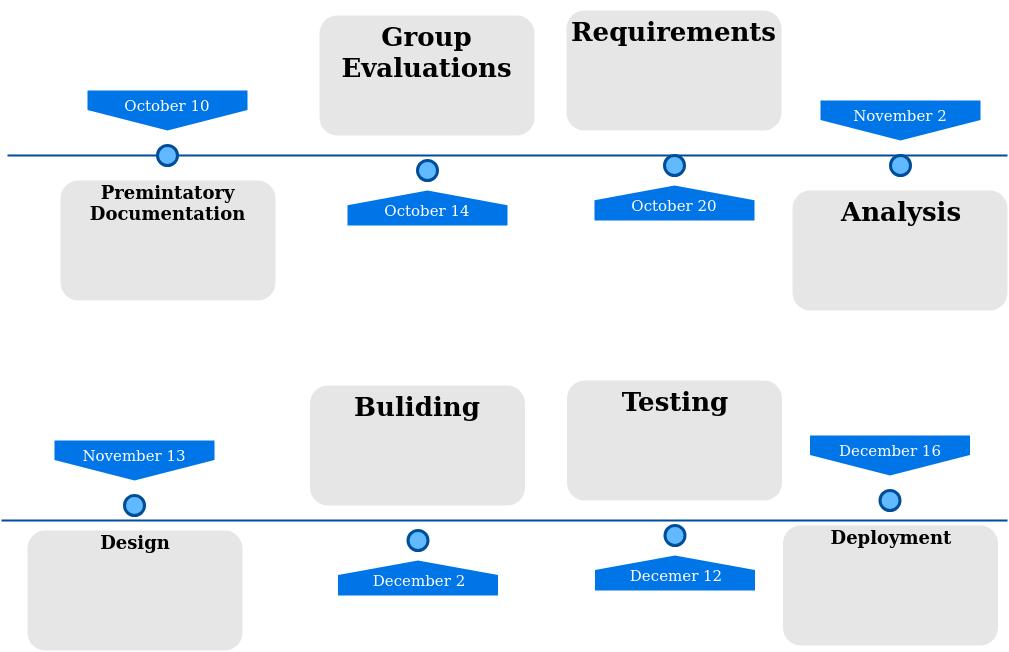
Framework: Waterfall

The process model, framework activities and task set that have been selected for the project are presented in this section.

**4.2 Task network**

****

**4.3 Timeline chart**



**5.0 Staff Organization**

The manner in which staff are organized and the mechanisms for reporting are noted.

**5.1 Team structure**

The team structure for the project is identified. Roles are defined.

|  |  |
| --- | --- |
| Name | Position |
| Zakariya Ahmed | Senior Tech Lead/SPMP manager |
| Cameron Labut | Developer |
| Mostafa El Majzoub | Developer |
| Muaz Alhaidar | Developer |
| Kevin Longworth | Developer |

**5.2 Management reporting and communication**

Mechanisms for progress reporting and inter/intra team communication are identified.

* Discord for online text/voice communication
* In person group meetings
* One-to-one meetings with Tech Lead
* Github issues board
* Weekly status on team

**6.0 Tracking and Control Mechanisms**

Techniques to be used for project tracking and control are identified.

**6.1 Quality assurance and control**

An overview of SQA activities is provided. This is an outline at this point and will be used to create a detailed plan later in the project.

* Standards
  + Private variable format: \_variableName
  + Public variable format: variableName
  + No use of global variables
  + Each variable has a one (number may vary depending) line comment describing what it holds
  + Precondition and postcondition before every method (if applicable)
  + Thorough documentation
* Team Reviews
  + Meetings every week to talk about the previous week’s progress and the forward week’s work.
* Security Management
  + Encryption of data contained in servers.
* Risk management
  + If any of the members sees fit that a risk has a higher probability of occurring they would communicate to the rest of the team, the team lead will decide whether or not to initiate a mitigation plan.
  + Assessment of discovery of new risks.

**6.2 Change management and control**

An overview of SCM activities is provided on how changes will be handled – communication of a change, how decision made of approval, defer or reject change request.

Proposed changes can be submitted to the discord and/or talked over at team meetings. Changes that don’t affect other team members are allowed to pass, as long as the Tech Lead gives approval.

**6.3 Tools**

What tools will be used to control access and versioning of artifacts.

* Git will be used, as it’s the industry standard for version control.

**7.0 Appendix**

Supplementary information is provided here.