**CS673S16 Software Engineering**

**Team Alpha - Requirement Tracker X**

**Project Proposal and Planning Team Alpha**

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Role(s) | Signature | Date |
| Allen Tshudy | Team Leader Security Leader |  |  |
| Rand Charland | Backup Team Leader  Configuration Leader |  |  |
| Chris Willis | Environment and  Integration Leader |  |  |
| Yue Lei | Requirements Leader |  |  |
| Vikrant Sanghvi | Design Leader |  |  |
| Enda Peng | Implementation Leader |  |  |
| Shucheng Ge | QA Leader |  |  |
|  |  |  |  |
|  |  |  |  |

**Revision history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Change** |
| **1.0** | **Allen Tshudy** | **2/6** | **Initial adds** |
| **1.0a** | **Randy Charland** | **2/6** | **Added section** |
| **1.0b** | **Yue Lei** | **2/7** | **Added Section** |
| **1.0c** | **Vikrant Sanghvi** | **2/8** | **Added Section** |
| **1.1** | **Allen Tshudy** | **2/29** | Updated Risk Management section |
| **1.2** | **Allen Tshudy** | **4/12** | **Updated Monitoring and Controlling Mechanism section** |
| **1.3** | **Yue Lei** | **4/23** | **Updated proposed High level Requirements section** |

[Overview](#h.mcsf9msn0iua)

[Related Work](#h.rk5sw48znesh)

[Detailed Description](#h.fg3z0hpd4q9v)

[Management Plan](#h.ds8oyr75pnh1)

[Process Model](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.27177f40uci)

[Risk Management](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.a4oqwntk3mw)

[Monitoring and Controlling Mechanism](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.ywdoc2clc9yt)

[Schedule and deadline](#h.tadq5mb0pici)

[Quality Assurance Plan](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.72e1f4uawy2r)

[Metrics](#h.b2haznn3yyz2)

[Standard](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.vc72k6dweldv)

[Inspection/Review Process](#h.f1c69ifi68h7)

[Testing](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.r5d5mhtlf0kq)

[Defect Management](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.54a4wuncjg1c)

[Process improvement process](#h.jhct37ebxxpn)

[Configuration Management Plan](#h.j5uvivmxqcsp)

[Configuration items and tools](#h.j9lrbekmwabv)

[code commit guidelines](#h.9jcvlqyba1ju)

[References](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.8mva2050iy7t)

[Glossary](#h.ty3i2nqffhtc)

# 

# Overview

The software development process begins with defining a set of requirements. These requirements describe in detail the features and functionality that the software product will have. Keeping track of requirements and their progress is a non-trivial task. Thus, a need is created for a piece of software that logs requirements and tracks their overall progress. This genre of tools is called Requirements Management software. They are intended to help software developers manage elements of their iterations.

3blueprints.com is a Requirements Management tool that has three components: a requirement tracker, bug tracker, and communication tool. The requirement tracker allows users to create stories in each iteration with priority levels, related tasks, difficulty of the story in hours, etc. The bug tracker enables users to keep track of bugs, assign remedying them to team members, and specify a priority for them to be fixed. The Communication Tool is simply used to keep teams in touch to work out issues related to all aspects of the software product development lifecycle.

# Related Work

3blueprints was developed by another team and handed off to our team, Team Alpha. Our goal is to apply bug fixes and new features to the software. We want to maintain 3blueprints’ distinguishing attribute of providing an all-in-one requirement, bug, and communication tool. Very few requirements management tools have an embedded communication tool. It is also entirely open source. It uses Python and the well-known Model View Template (MVT) design of the Django framework. This is a relatively simple design, which lends itself to being easily modified by future inheritors of the software.

There are countless requirements management software products. Here are three popular ones and how they compare to 3blueprints:

* Pivotal Tracker – 3blueprints provides a communication tool whereas Pivotal Tracker does not. Pivotal Tracker also has a more complex codebase that changes frequently. A blog post from 2011 states that they use a blend of JavaScript and jQuery, Prototype, YUI libraries. The post suggests that they may move to Backbone.js, Socket.IO, and node.js in future versions of the software. The back-end is built using Ruby on Rails.

* VersionOne – this requirements management tool is a Gartner Magic Quadrant leader. However, it uses a proprietary codebase and includes many features that may not be needed by most software development teams. The tool was built to support many software development methods.

* IBM Rational DOORS – this requirements management tool is very complex and includes its own programming language called DOORS eXtension Language (DXL). DOORS is proprietary product that was acquired by IBM and is now built using the IBM Jazz platform.

# Proposed High level Requirements

Existing features

1. Essential Features

* A new User should be able to sign up for an account.
* A user should be able to log in to the system if they have an account.
* An admin should be able to assign various roles for different users.
  + A user should be able to view only functionalities of his role.

i. Owner:

* + Owner should be able to view & create & edit & delete projects.
  + Owner should be able to change its role from owner to developer or client. Once the role is changed to developer or client, one can no longer change role.
  + Owner should be able to create & edit & delete user story in icebox. can add task in user story, set & change story owner, set hours, status, scale points.
  + Owner should be able to add & edit & delete comments
  + Owner should be able to create/edit/delete iteration
  + Owner should be able to add & edit & delete task in each iteration.
  + Owner should be able to upload & download documents.
  + Owner should be able to update status of story to started/completed.
  + Owner should be able to assign roles to users and add them to the project.

Ii. Developer:

* + Developer should be able to view projects
  + Developer should be able to create & edit & delete user story in icebox.
  + Developer should be able to add task in user story, set & change story owner, set hours, status, scale points.
  + Developer should be able to add & edit & delete task in each iteration, no matter who create the task
  + Developer should be able to add & edit & delete comments
  + Developer should be able to upload & download documents.

iii. Customer:

* + Customer should be able to view projects
  + Customer should be able to create & edit user story
  + Customer should be able to add & edit & delete comments

New features

* 1. Functional Requirements
     1. Essential Features
* New users can sign in the software management tool immediately if they have accounts
* Ability to assign users different levels of access to projects and iterations
* Clicking on the name of a project should perform the same as Open
* In a project, users could create backlogs
* Stories could be added to current iteration, icebox or backlog
* Users could mark an iteration as current one
* Users could delete the attachments and add description for them
* After uploading attachments, Users have the ability to link/refer them to a specific story within an iteration
  + 1. Desirable Features
* Build logs for stories so that developers could view how and who change them
* Each story within an iteration should also have a summary listing of the tasks within the story
* Developers could submit stories or bugs for approval
* Contact List are needed under user list for the convenience of manager
  1. Nonfunctional Requirements
* Limit login of a particular username to one active session
* Database remodeling
* Merging the bug and requirement tracker
* Project Detail Page could show status change of iterations and stories

* 1. Implemented Features

i. Functional Requirements

* Upload attachment
* Show task list within an iteration
* Show project detail directly when clicking project’s name
* Download/ Delete attachment under story
* Add issue list to requirement tool
* Create backlog views and templates
* List stories in queue by priority in backlog
* Add story type
* Add attachment feature to story level

Ii. Non-Functional Requiremnets

* Constrains on iteration date
* Each story has a work status displayed
* Remodel database
* Improve project detail usability
* Implement HTTPS
* Constrains for story & project level attachments

# Management Plan

# (For more detail, please refer to SPMP document for encounter example)

## Process Model

We will apply agile development method for this project. First we confirm with each other about which part we should work on and then we give a list of our total workload. We plan to have three iterations for the project. For each iteration we give the acceptable criteria. During each iteration, everyday our code should be runnable and we provide the demo at the end of each iteration. At April 27th ,2016, our team will be able to provide the final demo.

## Objectives and Priorities

Our goal is to make 3blueprints a really versatile product that can help developers manage their product, contact with each other, track requirements and bugs easily. For our team, we will focus on requirement tracker part. More in detail, our team will first fix existing bugs in the system and then add new features.

## Risk Management (need update constantly)

## 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Title** | **Estimate Likelihood of**  **Occuring**    **High, Medium,Low** | **Estimate Impact**  **High, Medium,Low** | **Priority**  **1 being the highest** | **Retirement Plan** |
| 1 | Losing personnel (person drops the class) | Low | Medium | 7 | We lost one person before the iteration started. We do not see this occur in the future. |
| 2 | Technical background  Most of the team is not familiar with Python, Django, GIT, linux  Or the testing frameworks that are used | High | High | 2 | We spent the our first iteration on learning the tools and framework. |
| 3 | Time management.  Some have full time jobs and others have classes | High | High | 1 | This is ongoing. we will hope to identify issues early and address them as soon as possible. |
| 4 | The team takes on more than they are capable of handling. We don't know what we are capable of. | Medium | High | 3 | The second iteration will be the bigger iteration. We will increase the story count. We will make sure we have good status to identify the risk early |
| 5 | Working on an existing project instead of creating a new project.  The main issue is not knowing how the product was original designed | Medium | Medium | 4 | We will try in introduce some new features in the next iteration. This will allow the team to design instead of fixing issues. |
| 6 | The lack of time spent on planning | Medium | High | 5 | We hope to have a better understanding of the application and have the ability to implement new features and fix bugs. |
| 7 | Additional homework for the class | High | High | 6 | We hope the lever of extra homework and presentation do not increase. We need to take this into account when we plan the next two iterations. |

## Monitoring and Controlling Mechanism

1. At least two meetings will be held each week
2. Face to face meetings will be held a half hour before class.
3. Call-in status meeting during the week.
4. Primary communication for the team is through slack or BU email address.
5. PivotalTracker to enter user stories and tracker progress.
6. GIT repository will be used as our source control system.
7. Continue to update the weekly report.

## Schedule and deadlines (need update constantly)

|  |  |  |
| --- | --- | --- |
| **Iteration 1**  **(Feb 10 - Mar 3)** | **Iteration 2**  **(Mar 3 - Mar 30)** | **Iteration 3**  **(Mar 30 - April 27)** |
| Fix a number of defects and usability issues  add new tests (unit tests and selenium)    Each member should a least be able to  fix one minor defect and add a new test.    We will need to learn to estimate correctly  so we now what we can accomplish as a team | Add a new feature and fix other usability issues  add tests for the new feature.    This will contain issues that did not make the first iteration.  If we feel we can add a feature we must do it in this iteration | Fix any defects with the new feature.  Finalize the feature with more tests |

# Quality Assurance Plan

# (For more detail, please refer to SQAP document for encounter example)

## Metrics : We will be using two metrics to keep track of quality of work we are doing. Product and Test metrics and Process metric to look for the process we are using is effective.

* + 1. **Definition**

**Product and Test Metrics**

**Complexity:** complexity of the project is determined by the number of features enhanced or fixed, number of line of code(LOC).

#### **Size and Test:** The size of the project will be measured in terms of bugs fixed and improvements implemented.

#### **Cost:** Cost of the project is determined by the man-hours put by the team on entire project. Man- hours are tracked by the all team members in weekly report document which specifies how many hours are spent on particular task.

#### **Coverage, Bugs and Fixes:** Test Case Coverage Rates (number of modules/classes/methods covered by at least one unit test ), Test Case Pass/Fail Rate will be determined in product and test metrics.

**Process Metrics**

1. Deadlines for each iterations along with number of task decided (estimated completion time vs actual)

2. Resource usage and allocation(estimated hours to complete vs actual)

* + 1. **Results** (to be completed at the end of each iteration)

Iteration 1

4 bugs were logged in pivotalTracker

3 bugs were logged in Slack

Total of 7 bugs where fixed

Iteration 2

3 bugs were logged in pivoltalTracker

8 bugs were logged in Slack

Total of 11 bugs where fixed

3 refactors

Iteration 3

5 bugs were logged in Slack

Total of 5 bugs were fixed

1 refactor

Code Review and code test report could be seen in our team project folder

## Standard

## Documentation standard:

## All the documents should keep the following properties:

Objective : the purpose of the document

Standard and Process :

defining the rules by which the individuals or assets within the scope must adhere; the process provides greater detail on the standard by describing how the individuals or assets comply with the standard.

Scope : identifying to whom to what assets the standards and process apply

Effective Date : date the document was implemented and enforced

Last Reviewed Date :

date the document was last reviewed for changes, updates, or document retirement.

## Coding standards:

1. suitable for repetitive use. A major requirement of a code or standard is that it can be used time and again. If a set of requirements is so specialized that it cannot and will not be applied repeatedly, it is not a code or standard.
2. enforceable. A standard's requirements should be worded so that a person auditing its use or application can point out where it has or has not been followed.
3. definite. Requirements that are too general or contain vague applications instead of workable instructions are impractical and often useless. Requirements should be expressed as specific instructions and never as explanations.

## Inspection/Review Process

Review: A process or meeting during which a software product is examined by a project personnel and other group member who participated the project.

A review process would be launched twice times before an iteration is finished, in order to evaluate its technical content and quality.

Our review process could be functioned as “software peer review”, every programmer in this process could participate the review process if it is possible. If the time is limited, at least the subgroup leader and the program owner should present the review process.

## (e.g. describe what are subject to review, when to conduct review, who do the reviews and how ?)

## Testing

i. **Unit Testing** Scripts using unittest (PyUnit) Module. Unit test scripts will be run on each build to update Reporting by each team member.

ii. **User Acceptance testing(UAT)**: at the end , we will test if all the requirement proposed initially are met which will be tested by real unbiased audience in real env.

A separate document about testing result should be linked here.

<https://docs.google.com/document/d/1i7Essck4VkmIyCdIswym4LlhzNV__h9qBMJJxncvXn4/edit>

## Defect Management (e.g. describe the criteria of defect, also in terms of severity, extend, priority, etc. The tool used to management defect, actions or personnel for defect management)

**Criteria of defect :** Defects are any error that results in failure to conform to specification or failure to operate properly or function efficiently.

Defining defect involved four specification along : category it belongs to, severity, priority and status.

1. **Defect category type:** **Software engineering process defect :** (involves design flaw, testing flaw, sdlc phase anomalies,documentation error , standard incompliance, flawed requirement document). **Configuration defect :** defect involving software patch flaw or upgrade flaw of tool used in project which are installed , administered, compile or resourced or misconfigured in a manner resulting in improper system operation, introduce vulnerabilities and degrade system performance. **Hardware defect(**defectsrelated to any hardware device like desktop,etc**),** **Code Defect(**flaws in the code implemented**)** and **Undetermined defect(**defect whose prime cause or nature is not known**)**
2. **Defect Severity :** Defect severity is classified according to their impact: fatal(blocks the current task), major(flaw in prime functionality of system causing system failure), minor(not causing system failure but flaws in functionality of system) and cosmetic(which does not hamper the any functionality of system). To determine severity we ask ourself few question regarding the defect such as is it blocking the system functioning? is system able to recover on its own ? Answers to these question determines the level of severity.
3. **Defect priority :** It’s different from severity and not related to it. Priority is relative importance of addressing and resolving the defect. levels are 1. critical , 2. essential and 3. cosmetic.
4. **Status:** the current status of the state of defect (open, closed or reopened)

**Defect Management tool for reporting :**

1. GitHub

Regressions and Deployment Bugs

2. Pivotal Tracker

New requirements that arise as a result of bug or issue

3. Slack

To get noticed by the individual for the defects encountered by him during unit testing.Discuss ideas and reach out to team members on any problems for help

# Configuration Management Plan

## Configuration items and tools

The client development environment will consist of the software below:

|  |  |
| --- | --- |
| **Software** | **Version** |
| Eclipse | 3.8.1 |
| PyDev Eclipse plug-in | 4.5.4.201601292234 |
| EGit Eclipse plug-in | 4.2.0.201601211800 |
| Python | 2.7.6 |
| Django | 1.6.1 |
| VMware Workstation/Fusion | 12.1.0/8.1 |

3blueprints is developed using a standardized Ubuntu 14.04.3 LTS virtual machine (VM) template. Software installed on the VM is listed in the table above. The VM template can be found as an OVF (Open Virtualization Format) on the class Google Drive share under our team’s folder.

In addition to the client development environment, Github will be used for revision control and the repository will be located at:

<https://github.com/ceyeclone/team_alpha_project>

Access to the repository will be controlled utilizing Github registered usernames and passwords.

## Change management and branch management

For each iteration, a branch will be made for each major development effort for that iteration that corresponds to a development sub-team. During the final week of the iteration, the branches will be merged with the master branch after each is reviewed by an agreed upon code review team for the spiral. Each development branch will be named using the following naming standards:

devteam<#>\_iteration<#>

or

devteam<#>\_iteration<#>\_<feature\_name>

## Code commit guidelines

During an iteration, development sub-teams should only commit to the development branch they are currently working within for that iteration. Once changes have been committed, a development teammate shall make a clean clone of the branch to ensure the code commits occurred correctly and the application functions properly.

Only the configuration manager or delegate shall merge or commit to the master branch.

All code commits should include a verbose description of the feature added or the bug fixed in the commit remark file.

# References

<https://blog.pivotal.io/labs/labs/what-powers-pivotal-tracker>

<https://www.versionone.com/product/>

<http://www.github.com>

<https://www.atlassian.com/git/tutorials/setting-up-a-repository>

<http://dhs.pa.gov/cs/groups/webcontent/documents/document/p_031767.pdf>

Glossary