Project Plan

Document Version: 1.0

Creation Date: 08/06/2014

Prospective Project Management System

Sam Reid, Khanh Hoang, Linsheng Wu

Table of Contents

1. Introduction

The Prospective Project Management System (PPMS) is a web-based system that will be developed according to this document. The system will be available to prospective Industry partners to submit project proposals to the Software Engineering Group Project (SEGP) course administrator for approval, and manage the project life cycle up until the point where it is assigned to a group of students.

2. Project Overview

2.1 Background

Currently a large amount of time is being spent on locating viable Bachelor of Engineering (Software) fourth year group projects. It is both difficult to initially locate interested industry representatives, due to low visibility of the program. Once a prospective industry partner is in communication, much time is spent obtaining the initial project idea and further time is spent communicating between the industry partner and the course coordinator to determine whether the project is viable. Projects must be achievable for three students to undertake in the allotted time or approximately 26 weeks. It has been estimated that the process of project idea refinement and validation currently takes on average 40 hours per project. Currently all the information gathered about prospective projects is not easily sortable and locatable (hidden away in coordinator email archives etc.). This can mean that some information may not be communicated about the project when assigning a project to a team of students.

2.2 Aims

The PPMS aims:

- To build an industry partner accessible system that helps the course coordinator to obtain Industry projects from the community passively (i.e. without having to proactively search).
- To manage all aspects of the project starting from the first point of contact with the initial idea through the approval stages, until the point where the project is assigned to a team of students.
- To reduce the overall time taken to source and approve Industry projects.
- To consolidate all gathered information about prospective projects in single place, allowing for better management and review.

2.3 Scope

Included:

- Final source code
- Correct versions of any any addons or extensions used.
- Database system schema setup
- Documentation and 'How-To' instructions
- Database schema description

Not Included:

- Ongoing system support

The success of this project from a use point of view will largely depend on the other Cluster C project: Past Projects site. This is because the PPMS needs high visibility to industry partners and the Past project site will play a large role in advertisement of the program. However, most of the development process can occur in parallel and isolation from this project, with minimal overlap. Indeed, the only shared components between projects is the inclusion of a shared style-sheet for site continuity.
No other projects are directly impacted by this project.

2.4 Operational Policies and Constraints

Due to the IT Services support policy for all new software developments, Ruby must be used as the application code language, specifically with Rails as the framework. We will also require the web-based application to be publicly accessible outside of the University internal intranet.
Initial development will be undertaken on an internal university unix server before being transferred to IT Services for deployment.

The project must also be finished prior to the completion of the University year, as the project is required to be analysed and marked as per the SEGP1 course curriculum.

2.5 Description of Current System or Situation

Currently this system is an entirely new development. There is NO system currently in place that does the same job. This means that no purpose built components are available to salvage from any previous system, with the exception of the central University login component.

Currently the task of procuring projects is undertaken by the course coordinator via email with personal and professional contacts. Currently the course coordinator (in past years this has been Kevin Maciunas) uses personal and professional contacts from their time in the industry to seek out company representatives and ask if these companies have any projects that need to be done. The main focus is negotiating a project that is the right size and complexity for fourth year students and that can be completed in the allotted time.

2.6 Users or Involved Personnel

Customers:
- Course Coordinators

Users:
- Course Coordinators
- Management Staff
- Industry Partners
- Bachelor of Software Engineering students

Development Team 7:
- Sam Reid
- Linsheng Wu
- Khanh Hoang

2.7 Support Concept

The system will NOT receive any ongoing support from the Development team listed in section 2.6. Due to other commitments and lack of funding, the system will receive limited support from the University's internal IT Services.

Version Support:
As this is a web-based SaaS application, the client will need a web browser installed. The PPMS will support Internet Explorer v10+, Mozilla Firefox (v1.0+), Google Chrome (v1.0+).

3. Development Background/Approach

The PPMS will be developed in a small 3 person team using the Agile Software Engineering process, coupled with the Scrum framework. All member of the development team (see section 2.6) will rotate through the following roles:

- Product Owner - Responsible for representing the views of the customer/client as well as liasing with them. i.e. the University/Course Coordinator
- Scrum Master - Responsible for managing the team and removing any obstacles blocking the projects progress
- Developer - Responsible for developing and testing components of the PPMS

The PPMS will be designed without a target deployment platform requirement. i.e. it will be able to be deployed on a Unix, or Windows based server. The only requirement is that all COTS components are installed with the correct versions.
As this is not a native application, any host with a web-browser listed in section 2.7 will be able to interact with the PPMS.

Our primary development work will take place on an internal University server, which will be configured in accordance with our Configuration Management plan. (See referenced Documents).

Due to the limitation that the development server we have been provided for development is only accessible within the university network, the majority of the development will be done on local machines in the form of separate modules. This component based design also works well as our team is essentially geographically dispersed, for development purposes, however we will get chances to meet.
At the end of each sprint, all components will be gathered together and joined on the university server, for integration testing to start (See Testing Plan, Timeline Gant Chart).

3.1 System Integration & COTS

Current effort required for this project is not entirely known. As such an incremental development process has been decided apon in order to ensure a workable prototype will be delivered. Another way will mitigate risks (also see section 9) is to use COTS components to hasten delivery.

Essentials:

- MySQL - SQL database system (Persistence tier) for storing project data
- Ruby - Server side application programming language
- Rails - framework for quicker implementation

3.2 Key Contacts and Stakeholders

Name	Role	Contact
Kevin Maciunas	Course Coordinator/Client	kevin.maciunas@adelaide.edu.au
Sam Reid	Group 7 - Rotating Role	samuel.reid@student.adelaide.edu.au
Lingsheng Wu	Group 7 - Rotating Role	linsheng.wu@student.adelaide.edu.au
Khanh Hoang	Group 7 - Rotating Role	buikhanh.hoang@student.adelaide.edu.au

4. Features, Primary Deliverables, and External Commitments

4.1 Feature List

Brief Description of Features:

- Collect initial project data
- Simple collection mechanism
- Data security
- Simple representation of project data
- Consolidated project data storage
- Simple data handling techniques
- Auto-emailing (push notifications)
- Send projects back for re-review
- Assign project to group
- Auto send project information to student development group

See Project Requirements Document for full details.

4.2 Customer Deliverables

Deliverables:

- Project Code Base
- Database Schema
- How-To Documentation + Built In
- Configuration Management Guide
- Installation Guide
- All project documentation for marking purposes

5. Project Schedule

5.1 Major Project Milestones

Date (DD-MM-YY)	Milestone/ Event	Entry Deliverable & Criteria	Exit / Notes
20-06-14	Planning Documentation release	Current Process/Requirements gathered	In progress
29-08-14	Prototype 1 Release	Planning complete	
12-09-14	Prototype 2 Release	Prototype 1 Must be Complete	
26-09-14	Prototype 3 Release	Prototype 2 Must be Complete	
10-10-14	Prototype 4 Release	Prototype 3 Must be Complete	
24-10-14	Prototype 5 Release	Prototype 4 Must be Complete	
07-06-14	Final Release	All documents/prototypes complete	

5.2 Detailed Project Schedule

See attached time sheets and Gant Chart for more detailed view.

5.3 Project Status Tracking & Working Meeting Minutes

As part of the Agile/Scrum process, regular meetings will be held to discuss current progress and any road blocks during the project life-cycle. The following information will be recorded for each of these meetings:

- Meeting attendees
- Meeting date/time
- Meeting length
- Discussion items
- Risks
- Issues
- Prototype status
- Deliverable status
- Open forum

6. Project Work and Product Estimates

6.1 Cost Estimate Summary

As this project is not-for-profit, and university based, there is no cost summary. We are using free COTS components and the physical hardware for the system is based on the current university setup which is not being paid for directly. All development tools are also freeware, unless trial period software is to be used.

7. Project Resource Requirements

7.1 Staffing/ Skill Requirements

As per section 3 and 12, roles will be rotated throughout the development process. As such, potentially not all members will have the same skills and skill gaps.

Role: Scrum Servant Leader.
Critical Skills:

- ➢ Comprehensive technical knowledge:
  - ○ Knowing how fix common technical difficulties.
  - ○ System configuration/Installation skills.
  - ○ Critical set of developer skills.
- ➢ Understanding Agile's best practices.
- ➢ Communication skills.
- ➢ Planning, facilitating, teaching skills.
- ➢ Ability to resolve conflicts in teams.

Skill Gaps: will be identified in sprint sessions, and be shorten after each iteration passed.
Role: Product Owner
Critical Skills:

- ➢ Communication skills.
- ➢ Market, business analysis.
- ➢ Understanding customer, team's capacity.
- ➢ Critical thinking, rational judgement.
- ➢ Good vision of what is needed to build.
- ➢ Availability

Skill Gaps: will be identified in sprint sessions, and be shorten after each iteration passed.

Role: Project Developer
Critical Skills:

- ➢ Installation and configuration of Rails environment.
- ➢ MVC Rails framework.
- ➢ Database migration.
- ➢ Testing with Rails predefined test framework, Rspec, Cucumber, etc.
- ➢ Basic Rails validations.
- ➢ Rails stack - callback, filters, plugins, engines, gem rack
- ➢ Active record associations.
- ➢ Basic HTML, CSS, and JavaScript.
- ➢ GIT – clone, commit, and push.
- ➢ Design pattern
- ➢ Object orientation
- ➢ App deployment to production environment

Skill Gaps: will be identified in sprint sessions, and be shorten after each iteration passed.

7.2 Plan to Fill Skill Gaps

- **Skill gaps prioritization (part of sprint session):**
  In sprint session, specific missing skills of each team member will be prioritized based on the needs of his roles in incoming iterations. After the seson, we will come up with a list of skills in critisical order for each specific team members to take care of. The list does not need to be the whole set of skill gaps listed above, but merely a list of what skills are needed for the current situation.

- **Iteration-based learning material gathering:**
  Each team member will be responsible for looking up and recording useful learning resources in correspondence to his pre-specified skill gaps for each sprint. These learning materials can be (e) books, video/plain text tutorials, online courses, or related articles, etc. This information will be well organized in a document accessible to everyone, making the training process easier.

- **Training and in-team Demonstration**
  <u>Training</u>
  Each team member is responsible for quickly filling his skill gaps with the assistance of learning materials, teammates and/or course lecturers/tutors/friends (if possible). Specific learning activities of this training can be taken through:
  - Self-study
  - Pair(or team) programming
  - Pair(or team) learning – if multiple team members have the same kill gap and time allows.
  - Consultation with both internal and external sources.

  <u>In-team demonstration</u>
  For each sprint, we will try to allocate an appropriate time for each team member to demonstrate what skill gaps they have filled and to what extent. Depending on the knowledge requirements of other team members, a demonstration on what/how they've learnt will be carried out in form of a short tutorial for the benefit of the whole team. Since the learning process takes time, this tutorial should be a guideline, rather than a detailed technical tutorial. If the time does not allow us to carry out a demonstration session, then an equivalent short report is needed for each iteration.

This session for every sprint is critical to make sure the training process of team is in consistent progress, catching up with the needs of the project.

7.3 Tools

Tools to be used in the development process. Note: These will NOT be shipped with the final product, unless specifically stated.

Purposes	**Tools**
User stories management	3-by-5 cards, Pivotal Tracker - subject to licensing
Source Code/Software Version management	Github
Code editor/IDE	Vim, Sublime Text 3 (with ruby plugins), TextMate, Gedit, Notepad++ / Komod, Aptana RadRails, Netbeans, RubyMine, phpMyAdmin - for database admin and query debugging
Testing	Rspec, Cucumber, Capybara, Selenium, ZenTest
Documentation	Rdoc (with Rake's help), Microsoft offices, Google Docs
Deployment	Capistrano

8. DEPENDENCIES AND CONSTRAINTS

8.1 Constraints

This project is subject to the following constraints:

1. Use of Ruby on Rails: The project must be completed, to a large extent, in the Ruby language coupled with the Rails framework in order to be compatible with the current university systems and receive support post hand over.
2. SaaS application: The project must conform to the Software as a Service ideology due to course requirements. However this is the obvious design choice.

9. RISK MANAGEMENT

9.1 Risk Management Strategy

Depending on how critical the risk is and its nature, we may choose either of the strategies below to deal with the risks:

- **Avoidance:** Analyze risks and identify all possible factors that may cause risks, by removing these factors we can avoid the possibility of the risks happening to our project.
- **Mitigation:** In this strategy, we can limit the impact of the risk on our project by either minimizing sources which leads to the rises, or setting up ways to reduce consequences that are possibly caused by the risks.

- **Transfer:** Pass risks on to others. They will be responsible for any effect of the risks that cause. But this is often likely to cost us somehow.
- **Acknowledgement:** If the risks are hard to deal with or impossible to handle in any way (either of by avoidance, mitigation or transfer) due to the current situation of our project, then accept the possibility that the risk may happen. This will give us more time to enhance the system in other respects.

9.2 Initial Risk List

This is the initial risk list; descriptions are provided below the table.

Risk number	Risk Priority (H, M, L)	Likelihood of Occurrence	Risk name: brief description	Mitigation Strategy
1	M	Likely	Not knowing ruby in time for project development start	MITIGATED
2	H	Likely	Not finishing on time	MITIGATED
3				

9.2.1 Risk Details:

1. A poll of the development class revealed that none of the developers had used the Ruby on Rails bundle previously, so it is likely that there will be some knowledge gaps when developing entirely with RoR. This risk is being mitigated by learning Ruby on Rails during the holidays by all members of group 7.
2. There is a risk that the entire project process will not be completed on time due to other university commitments and unknown hazards. This is being mitigated by the use of extensive planning and progress tracking. This risk is further mitigated by prioritising features to ensure the system is usable and works before adding surplus features.

10. Project Configuration and Data Management

10.1 Configuration Management

This project team has been provided a Linux server by the University of Adelaide's IT Services team. Initially the server has not been configured for our project use and needs to have all components listed in section 3.1 installed. This will be done in the configuration management stage (see section 5.2/Gant Chart for more details) and will be documented in a separate Configuration Management document, however the basic outline is as follows:

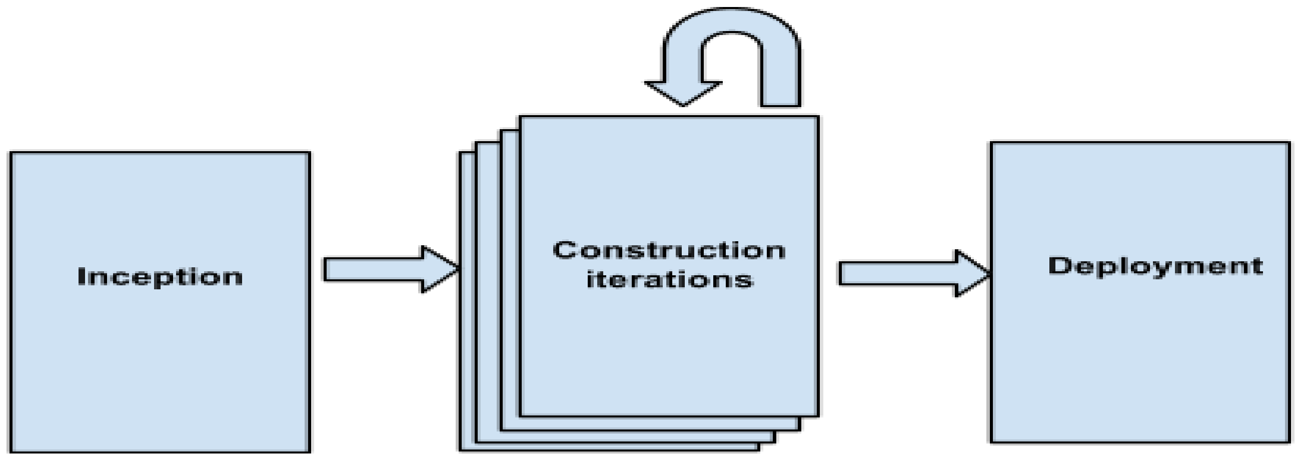
1. Install + Configure Ruby on Rails
2. Install + Configure MySQL Database
3. Install PHP if needed for #4
4. Install + Configure phpMyAdmin

11. Project Quality and Measurement Plans

System quality will be measured by how well the final project meets the gathered requirements. Extensive testing will be done, including a 'prospective user' BETA test which will help determine quality in a project which doesn't have an extensive customer base such as this one. The BETA testing stage is described in the Testing Plan, and similarly the Quality assurance is described extensively in our QA plan.

12. Project Process

12.1 Software Life Cycle Model



Inception phase:

- Building team – getting familiar with each other and planing basic activities ahead.
- Gathering core requirements for the project by working with customer and other teams.
- Creating project plans, test plans, QA plan and a prototype for the software system.
- Setting up facilities needed for project development, activities including:
    - Identifying specific tools, working environment we will be using throughout the project.
    - Installing and configuring those development environments.
    - Getting familiar to the temporary production server's environment to which project is finally deployed to.

Construction iterations:

- *Sprint planning meeting (4 hours):*
    - Product owner and team are working together to come up with an agreement on a set of user stories that team will deal with in the iteration. The decision-making on feature prioritisation is up to the product owner. However team members have power to raise concerns of any potential difficulties caused by that decision. After final agreement, product owner will add those stories to sprint backlog (Using Pivotal Tracker).
    - Product owner stays with team to decompose the stories into workable tasks. Teams will generate specific test plan and quality assurance criteria for this iteration's release (TDD+BDD). Additional workable tasks due to these planing, will be added to and organised on the backlog as well. Additionally, the role rotation and corresponding skill gaps needed for each individual in next iterations are also identified and a training plan is scheduled.
    - All other aspects of the project development specific to this print are identified and planed for a resolution (Risks, assumptions, constraints, dependencies, tools, etc).
    - Defined tasks will be assigned to groups of team members (1, 2, or 3) together with delivery time deadline.
- *Ready to work - activities: includes:*
    - Designing system models/structure and tests
    - Writing tests.
    - Implementing current features.

- ○ Inspecting Code and fixing any defect detected.
- ○ Integrating features with each other or with intern/extern-al system.
- ○ Conducting system tests and quality assurance and finally acceptance tests
- ○ Deploying the system internally.

- *Daily Scrum:*
  - In this 15 minutes daily meeting, each member report their progress, their next tasks, and obstacles they encountered.
  - Time is decided by Product Owner based on the availability of individual team members.
- *Scrum review and retrospective session*
  - Product owner reviews the backlog and indicate what were accepted and what were unfinished or rejected at the last second, and demonstrate detailed reason for those things.
  - Based on that team can reviews fully what were done well (how?), and what were failed to get done (why?) in the iteration. From the review, Scrum master can have better overall on what is likely to cause troubles to team in next iterations and plan some solutions to get rid of those likelihood in the futures. Additionally each team member have some discussion on what is needed to do better in next iteration.
- **Other meeting sessions**
  - Depending on specific demands of team in regard to the project, team may have some other urgent sessions to deal with those specific problems.
- **Meeting session with customer and other team:**
  - Working with customer and other team is an integrated part of agile process in general and of this project in particular. This is for gathering new user stories, resolving any problem/enquiry associated with customers and for integrating our system with that of other teams.

Deployment phase

- Deploying final version of project in integration with other system developed by other team in cluster to the temporary production server.

13. Referenced Documents

1. Detailed Requirements Document - version 1.0
2. Configuration Management Document - version 0.0
3. Project Schedule Gant Chart - version 1.0
4. Testing Plan - version 1.0
5. QA Plan - version 1.0

14. Glossary

COTS - Commercial off the Shelf
SEGP - Software Engineering Group Project
PPMS - Prospective Project Management System
SaaS - Software as a Service
RoR - Ruby on Rails

15. Change Record

Modified Time	Issue #	Description	Reason	Modifier
8/06/2014	Pre-release	Initial draft created	NA	Sam Reid
14/06/2014	Pre-release	● Action: Adding stuff for sessions: 2.6 , 7.1, 7.2, 7.3, 12.1,15 , 9.1. ● State: All in progress	Attempts of filling up project plans	Khanh
18/06/2014	Pre-release	Completed sections: 2.2, 2.3, 2.5, 2.7, 3.1, 3.2, 4.1, 4.2, 5.1, 5.2, 5.3, 6.1, 8.1, 9.2.1, 10.1, 11	Previously Incomplete	Sam Reid
18/06/2014	Pre-release	Finish up section 7, 12	7, 12 previously incomplete	Khanh