B.U.B.O.L.O. Software Project Management Plan

CS673 Clone Productions
1 March 2014

1.0 Overview - The Software Project Management Plan is the central document that highlights all of the plans to be put into practice during execution of the B.U.B.O.L.O. project. This document will also highlight the organizational structure and workflow planning for the team.

1.1 Summary of Project - B.U.B.O.L.O. is a clone of the 1987 game bolo. It is a 2-D, top-down tank explorer shooter. On the field, there are two tanks, several bases, pillboxes, various terrain types, and other varying objects. The objective for each tank is to take over all of the bases on the map.

1.2 Evolution of the SPMP - The SPMP will undergo changes when other plans are created or updated and reviewed monthly to ensure consistency with other plans.

Here is the revision summary:

Document Creation	1 March 2014

2.0 References

- B.U.B.O.L.O. Software Configuration Management Plan
- B.U.B.O.L.O. Software Quality Assurance Plan

3.0 Project Organization

3.1 External Interfaces - Our customers are the only external interface for the project team. However, it is the single most important interface.

3.2 Internal Structure - Our team is comprised of three evolving sub-teams of equal control. These teams are defined at the beginning of each sprint and set to execute a defined set of requirements for that sprint. Each sub-team leader flows work to the members of his/her team.

3.3 Roles and Responsibilities - The team is comprised of three distinct roles the

project leader, the subteam leader, and the team members.

3.3.1 Project Leader - The project leader ensures that all sprint requirements are clearly defined, the specific roles sub-teams are defined based upon the needs of the requirements, the defined requirements are properly flowed down to the individual subteams, and that the sub-teams are properly resourced to execute those requirements within that sprint execution cycle. The project leader also ensures that all plans and processes are followed appropriately across the teams in a consistent manner.

3.3.2 Sub-Team Leaders - The sub-team leaders (STL) take a high level overview of the tasks assigned to that sub-team and provide feedback to the project leader regarding any concerns. STL has a functional understanding of the available time and technical ability of each team member within the sub-team, and will work with the team members to divide up tasks according to volunteering and skill levels, while also maintaining balance of workload between team members. Each STL is responsible for their sub-team's branch being ready for integration in the Production branch, and must approve any requests from team members to add their individual code additions to the sub-team branch.

3.3.3 Team Members - Each team member is a member of a sub-team and is responsible for on-time completion any tasks for which they have volunteered, or been assigned. Team members are responsible for maintaining their own Individual branch in the repository and requesting merge their individual branch with their sub-team branch when ready. Team members are responsible for maintaining an in-depth knowledge of the sub-team they are a member of, and a functional understanding of the other sub-teams and project overall.

4.0 Managerial Process Plans

4.1 Project Startup Plan - The project formally begins after the identification for the need of the product. The particular need for the B.U.B.O.L.O. product was identified in a classroom environment as a need for a semester long project which was determined to be a video game. After the need was determined, the team was formed to discuss in further detail what was needed by a game as well as assign the project leader and STL's for the duration of the project. The team determined that a modern remake of the 1987 game Bolo was in order. After the concept was determined a requirements discussion took place to determine what requirements were needed for the game in order to fulfill the need determined at the beginning of the semester. These requirements discussions naturally led

the team into design and implementation of the B.U.B.O.L.O. project.

4.2 Work Plan - The team members have agreed upon meeting weekly to either set the requirements for the next sprint or to discuss sprint progress and design decisions. The STL's then take the design elements and assign the workflow to team members to execute the coding, commenting, and unit testing for those design elements. As documentation is required, a member of the team leadership will draft an outline version of the document with comments of what will be inserted into each paragraph. The creator of the document will then divide the writing of various parts of the document to various team members. The document is then due to be completed the Sunday prior to the document deadline where the team then has 48 hours for final review. During the review period, any team member may submit comments against the documents for the creator to adjudicate. Finally, the document is published once the creator has adjudicated all comments and finalized any changes within the document.

4.3 Control Plan - If any changes to the project requirements or sprints are identified, the changes are to be brought up at the weekly in-person meeting, or via electronic collaboration methods, for discussion by the team. Each team member is responsible to review the changes in a timely manner and provide any feedback they may have. The project leader and affected STL's are responsible for deciding what steps are required to address the change and inform the other STL and team members. Requirements documents will then be updated with the change, as well as any changes to development plans or schedules.

4.4 Risk Management Plan - The projects Risk Management Strategy revolves around systematically identifying, categorizing, and prioritizing all program risk. Risk management begins with observing the project status and requirements and empowering all team members to identify risks. After a risk is identified, the team discusses and agrees upon the probability the risk will occur and the consequence if the risk is realized. The risk is then mapped onto a 5 by 5 matrix based upon the consequence and probability to help with prioritization. Once the risks are prioritized they are tracked to identify changes in the status of the risk and reassessed for changes biweekly as well as factor in any new risk the team identifies. During this reassessment period, if a risk is fully mitigated it is then retired from tracking. However, if a risk is realized, a plan is put into place to make changes to the program to account for the impacts to the project.

4.5 Project Closeout Plan - At the completion of the project an executable with basic documentation (or in-game help file) will be available for use/demonstration and for the

customer to approve. A summary of the project's development timeline, all met/tested requirements listed, and an overview of the current development state of the software will be created. Any unmet requirements, or known bugs, will be documented. All documentation generated during the development process, including in-person meeting minutes and electronic collaboration, will be archived along with the source code and the above summaries for future reference and maintenance.

5.0 Technical Process Plans

5.1 Process Model - The B.U.B.O.L.O project will use the iterative approach of the agile software development model. The basic flow of the process will consist of 3 sprints with the overall work to be done for the project would be relatively equally divided between the sprints. Each of the sprints requirements are decided before the beginning of the sprint and as per the requirements and then the work is divided between the 3 sub-teams. The progress of the teams is measured throughout the duration of the sprint and compared with the goals that are set at the beginning of the sprint to avoid missing any set targets. The key concepts of agile development regarding the management, organisation and structuring of a small team will be applied. Simplicity is stressed. Potential management phase configurations will be touched upon in documentation only (if at all) given the scope of the class.

5.2 Tools - The following paragraphs are a list of tools that the B.U.B.O.L.O. team will use for the execution of the development project.

5.2.1 Eclipse IDE - As the project is based on the JAVA programming language eclipse was decided upon by all the team members as the IDE of choice. Release under the terms of the Eclipse Public License, Eclipse SDK is free and open source software (although it is incompatible with the GNU General Public License). The coding convention are documented as an eclipse preferences file that ensures that all the coding convention that are decided upon by the team are easily distributed to all team members for compliance.

5.2.2 Tiled - Tiled is a general purpose tile map editor. It is meant to be used for editing maps of any tile-based game. It is also very flexible, for example there are no restrictions on map size, tile size or the number of layers or tiles. Also it allows arbitrary properties to be set on the map, its layers, the tiles or on the objects. Finally, TIled offers the capability to save the maps as an easily translated JSON file format.

5.3 Infrastructure Plan - The B.U.B.O.L.O project team will use the following infrastructure to realise the goals of the project.

5.3.1 GitHub - It is a web based hosting service for software development projects using the git revision control system. It makes it easier to collaborate on software projects in a distributed environment. Also it has features for tracking bugs.

5.3.2 Google Drive Documents - It is a freeware web-based office suite offered by Google within its Google Drive service. It allows users to create and edit documents online while collaborating with other users live. We have used it effectively to collaborate on the Software Quality Assurance Plan (SQAP) and the Software Project Management Plan (SPMP).

5.3.3 Google Groups - It is a service provided by Google Inc. that provides discussion groups for people sharing common interests. Users can participate in threaded conversations, either through a web interface or by e-mail. We have used it for discussing various questions about the architecture and other details of the project, such as any specific questions the team members have for each other.

5.4 Product Acceptance Plan - This section describes the method of acceptance of the product deliverables by the customer. On the completion of every sprint, the customer will install the product and examine it. If the customer requests additional changes or improvements, they will be made in accordance with the change/control plan defined in section 4.3. An approval signature is required from the customer for final delivery of the product.

6.0 Supporting Process Plans - The following are a list of plans that are either already created or plan to be created by the team at some point throughout the development of the project. Each of the plans will briefly describe the overlying process of the respective plan in which further details can be obtained from that document.

6.1 Configuration Management Plan - The team’s configuration management plan divides the teams into manageable layers to control the configuration of the release product. The master baseline will be branched into a production branch which in turn will only be merged once a fully releasable sprint product is in place. The production branch will be the baseline from which all subteams will synchronize work by branching the respective sub-team’s branch from the production branch and merging back all approved updates.

Each team member will own a personal branch that is pulled from the branch of the subteam to which the member belongs. The member must request a pull from the team leader in order to merge their work back into the sub-team's working branch. A more detailed version of this process can be found in the B.U.B.O.L.O. team's Software Configuration Management Plan.

6.2 Verification and Validation Plan - TBD

6.3 Documentation Plan - TBD

6.4 Quality Assurance Plan - The team's quality assurance plan builds its foundations on the actions of individual members testing their individual work and submitting to the sub-team lead for further review and integration. Once the lead has determined that the submitted code does not induce errors into the team's branch it is then merged into the team's code. Once a sub-team is prepared to submit the code into the production branch, another lead reviews and test the code then approves or disapproves the merge into the production branch. For further details of this process, please review the B.U.B.O.L.O. Software Quality Assurance Plan.

6.5 Reviews and Audits Plan - TBD

6.6 Problem Resolution Plan - TBD

6.7 Process Improvement Plan - TBD