2014

Software Development Plan (SDP)

CMIS330

IAN NELSON

DEVELOPED FOR **JENEO K. DURHAM**

Table of Contents

1. Overview

The software development plan (SDP) provides an overview of the schedule, tasks, and resources required to build and deliver the Bed and Breakfast Management System (BBMS). This document will list, in detail, the project assumptions, deliverables, and schedule.

1.1 Project Summary

1.1.1 Purpose, Scope, and Objectives

The objective of the BBMS is to provide software to increase the efficiency of personnel managing small bed and breakfast establishments. By providing a software application to manage the calendar, available reservations, rooms and date ranges, user information, and financial transactions associated with reservations, the managers of the bed and breakfast can easily schedule reservations amongst a large set of customers, while easily managing reservations across a broad time-frame.

In order to accommodate both the needs of management and potential customers, the software should provide functionality specifically limited to the above areas. By not providing out-of-scope functionality, such as extended financial transaction holdings, information about other establishments, et cetera, and the scope of the software is specific to the needs of the customer.

The objectives of the BBMS are guided by efficiency. As a result, extensive integration with the customer will be required. Elimination of manual processes, where appropriate, are required to assist the user of the BBMS in working with customers over the phone or in-person, depending on the situation at hand.

The product for delivery should include:

1. Software package consisting of:
    a. Calendar management and reservation checks
    b. Guest reservation form
    c. User information form
    d. Financial transaction form for payment
    e. Polling checks for reservations with bad guarantees
2. User guide and manual
3. Quick use pamphlet

Successful delivery of these features is dependent on user acceptance testing for milestones where the software is versioned (alpha, beta, version 1.0), successful automated testing of all unit and integration testing across the source code repository, and guided walkthroughs of the user guide and quick use pamphlet.

A list of product requirements is available in the BBMS Software Requirements Specification, version 1.0.

1.1.2 Assumptions and Constraints

For the development of the Bed and Breakfast system, it is assumed that an information technology (IT) deployment exists in order for the software to be installed. This system is not provided with the BBMS, and is out-of-scope of the requirements.

Additionally, the development of the BBMS is designed around portability. By focusing on the use of language (Java) and minimizing the user interface requirements (text-based versus a graphical user interface), this software can be portable to almost all operating system platforms.

Assumptions for the BBMS are minimal due to the lean design process, and lack of reuse across the development cycle. Where appropriate, the development teams shall use free and open-source software (FOSS) to accelerate development, and make use of libraries for specific functions (such as database connectivity). In addition, the use of databases within the environment are dependent on the IT infrastructure in use by the customer. Testing for several variants of relational database management systems (RDBMS) will be required to assure compliance with these technologies, and will be referenced within the user guide deliverable.

Interfaces with other software and required acquisitions of software (FOSS-driven) are not applicable for the BBMS.

1.1.3 Project Deliverables

Project deliverables for the BBMS are focused on the following:

1. Software package consisting of:
    a. Calendar management and reservation checks
    b. Guest reservation form
    c. User information form
    d. Financial transaction form for payment
    e. Polling checks for reservations with bad guarantees
2. User guide and manual
3. Quick use pamphlet

The software delivery of the BBMS will consist of a Java application with executable shell scripts for Linux and Microsoft Windows. This software will also contain an example configuration file for database connectivity, showing configuration options for MySQL, PostgreSQL, and Oracle. The software will be made available for download electronically for customers wishing to receive digital media, as well as a version on CD-ROM for those requiring physical media.

The user guide and quick use pamphlet will be delivered via electronic download and CD-ROM as well, in the form of a PDF. Delivery to the customer, whether electronic or CD-ROM, will also include one (1) hard paper copy in a binder, with category separation and labeling.

2. References

The BBMS project management plan consists of the following documents:

1. Software Development Plan, version 1.0 (this document) – Ian Nelson
2. Software Requirements Specification, version 1.0 – Ian Nelson
3. Software Design Document, version 1.0 – Ian Nelson
4. Software Test Specification, version 1.0 – Ian Nelson, Christy Gill, Hector Santiago, Jessie Wasilewski, and Ed Russell.

3. Definitions

The following definitions and lexicon are referenced within the BBMS design and development plan:

1. BBMS – Bed and Breakfast Management System
2. Agile – A lean development methodology used by the BBMS team to manage requirements on an iterative basis. The opposite of waterfall.
3. Waterfall – A full project management and development methodology used to describe the process and workflow in a linear and fully resolved fashion. The opposite of agile.
4. Outreach – The process and business analysis associated with a project lead or assigned specialist working with customers to determine requirements, differences in opinion regarding implementation, and ensuring successful delivery of the software product.

4. Project Organization

This section delineates how the project team is organized for BBMS, both in external and internal interfaces. Overall, the project team is designed to be small in nature, and accommodate for efficiencies with communication structure between the team itself internally, as well as customer outreach to accommodate and validate requirements.

4.1 External Interfaces

Due to the small and efficient design of the BBMS development team, external interfaces other than directly to the customer are not applicable (N/A).

4.2 Internal Structure

The BBMS development team structure shall consist of two (2) Java developers, a user experience (UX) specialist, program manager, and customer outreach liaison. This team composition will create a structure allowing for direct development of the software requirements, design and modification of the user interface elements as agreed upon between the UX specialist and the customer, and requirement and process management between the team and customer base via the program manager and customer outreach liaison.

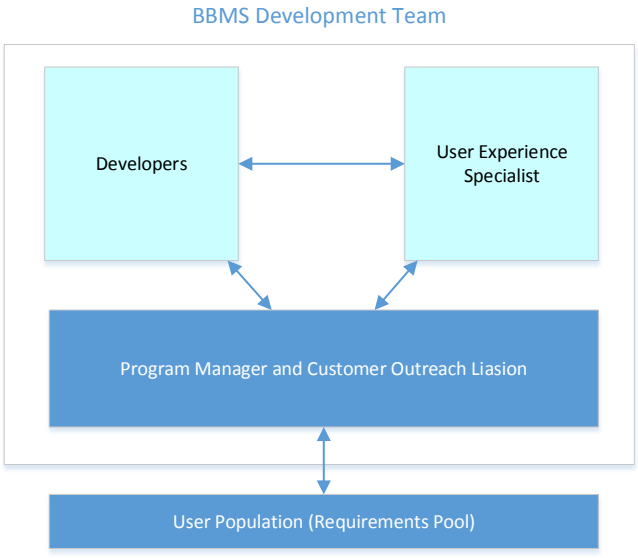

Figure 1. BBMS Development Team Structure and Process Flow

As seen in figure 1, the BBMS development team is designed to encapsulate and isolate process needs specific to team members, while allowing for maximum flexibility and communication. In this diagram, the developers and UX specialist are highlighted in light blue. These members perform the vital function of the software and interface construction directly. As a result, some shielding and isolation from the main process should be given to allow for maximum efficiency. This is done through communication directly to and from both the program manager for high level process and requirements management, as well as the customer outreach liaison to ensure the customer's requirements are being created with the user's wishes in mind. Additionally, this two-way flow ensures that communication between each portion of the team exists for optimum results.

As stated above, isolation is key to the efficiency of developers and designers. While this is an important fact to state, it is also critical to ensure a constant feedback look to and from the customer to ensure the requirements are being met in a satisfactory fashion. To guarantee this process, high level customer engagement milestones are handled by both a program manager and customer outreach liaison, but more importantly are the internal or low-level engagement milestones. These are conducted specifically by the customer outreach liaison directly with the customer on a frequent interval. This ensures that customer requirement questions and issues from the development team are handled directly and frequently, without having to stop developers from working, and providing a single point of contact with the customer.

All software within the BBMS development cycle will make use of software version control, specifically implemented with Git. Developers will ensure the following requirements are met for change management and version control to ensure quality assurance across the software line:

1. Source code is committed to the Git repositories at least once per 24 hours, and more frequently as able.
2. Commits to the source code repository must only occur after sufficient testing has been completed to verify that the changes work, and do not break builds.
3. Test infrastructure will make use of automated change management tools, specifically using Ansible. This will allow repeatable results for the creation of virtual environments that builds are deployed to, and remove team requirements for system administrators.
4. All source code changes and infrastructure change management scripts will be committed to Git when available and stable.

To ensure that software has been tested and provide a high-level of quality assurance, developers are responsible for providing complete unit and integration test suites for automated testing. These tests are to be built in JSUnit to match the Java class design (See Software Design Document (SDD) for additional details), and accurately match functionality at the method level during development. Specifically, the Git commit process should have a post-commit hook to run automated tests which will force a 100% pass rate of code moving into the source code repository.

Where external interactions occur, such as database configurations, integration tests should be created to test this functionality. For supported databases, each configuration should have an appropriate integration test which passes and directly corresponds to a requirement and statement of configuration within the user's guide.

Additionally, all automated test suites for unit and integration testing should match a corresponding requirement within the Software Test Specification (STS), and be committed and version controlled within the source code repository.

4.3 Roles and Responsibilities

The BBMS development team is constructed of the following work roles and responsibilities:

Two (2) Java developers – Performs software development tasks using the Java language. Must be fluent in the Java language, as well as libraries and methods used to implement software in a cross-platform process. Additionally, the understanding and configuration of data sources and connectivity to persist data in the BBMS will be required to validate and test requirements associated with saving data.

One (1) user experience (UX) specialist – Performs the task of design, design efficiencies, and special internal/external requirements associated with the human/computer interaction (HCI) of the BBMS. This person shall provide mockups based on user engagement and customer interaction to the development team for the user interface, as well as design and refactor designs related to all elements of the user interface.

One (1) program manager – Performs the task of project oversight, requirements management, and schedule obligations for the team. Will deliver project-related integrated master schedules (IMS) as well as conduct schedule conflict resolution to mitigate any risks associated with the waterfall methodology. In addition, this position is also responsible for feedback from the customer base, and providing on-time deliverables at periodic milestones within the project lifecycle.

One (1) customer outreach liaison – Performs the task of centralized interaction with the customer for consistent and constant feedback through the project lifecycle. Also serves as the interface to the BBMS development team to create feedback to the customer when required. This position is key in ensuring not only successful delivery of the software package and all other associated materials at regularly scheduled intervals as defined by schedule milestones, but also to make the customer feel comfortable with all stages of the process.

5. Managerial Process Plans

This section of the Software Development plan highlights the work schedule and task structure required to meet the customer needs, requirements, and schedule as agreed to by both parties.

5.2 Work Plan

The following section discusses in detail the work breakdown schedule and project schedule for the BBMS development team.

5.2.1 Work Activities

The following table constitutes the full breakdown of tasks under the master Work Breakdown Schedule.

Task Name	SDLC Task Category	SDLC Sub-Task
Form Teams	Analysis	Business Analysis
Initial Customer Engagement	Analysis	Business Analysis
Identify Customer Needs and High-Level Requirements	Analysis	Software Analysis

Develop Milestones	Analysis	Business Analysis
Define Budget	Analysis	Business Analysis
Develop Requirements	Design	Software Analysis
Deliver and Discuss Software Requirements Specification	Analysis	Business Analysis
Software Design Review	Design	Software Design
Create Software Design Document	Design	Software Design
Deliver and Discuss Software Design Document	Analysis	Software Design
Create Software Testing Specification	Design	Test Design
Deliver and Discuss Software Testing Specification	Analysis	Business Analysis
Build Calendar Lookup Services	Code	Service Development
Build Calendar Lookup UI	Code	Interface Development
Build User Information Services	Code	Service Development
Build User Information UI	Code	Interface Development
Produce Initial User Documentation	Code	Documentation Development
Build and Package Alpha (v0.1)	Code	Software Build
Release 0.1 (Alpha)	Code	Software Build
Initial User Acceptance Engagement	Test	User Testing
Build Room Reservation Domain Object	Code	Data Development
Build Room Reservation Services	Code	Service Development
Build Room Reservation UI	Code	Interface Development
Build Payment Domain Object	Code	Data Development
Build Payment Services	Code	Service Development
Build Payment UI	Code	Interface Development
Build Guest Reservation Domain Object	Code	Data Development
Build Guest Reservation Services	Code	Service Development
Build Guest Reservation UI	Code	Interface Development
Produce Extended Functionality User Documentation	Code	Documentation Development
Build and Package Beta (v0.5)	Code	Software Build
Release 0.5 (Beta)	Code	Software Build
Mid-Point Customer Engagement	Analysis	Business Analysis
Mid-Point User Acceptance Engagement	Test	User Testing
Build Reservation Scanning Services	Code	Service Development
Build Database Connectivity Services	Code	Service Development
Produce Final User Documentation	Code	Documentation Development
Build and Package Final (v1.0)	Code	Software Build
Release 1.0 (Final)	Code	Software Build
Final Customer Engagement	Analysis	Business Analysis
Final User Acceptance Engagement	Test	User Testing
Delivery of Product (BBMS v1.0 and Documentation)	Code	Software Build

Figure 3. Master Work Breakdown Schedule

In this schedule, the overall tasks are mapped to the calendar schedule in Figure 2, but are assigned by category in the software development lifecycle (SLDC) and their corresponding sub-task category. This corresponds to the full breakdown of work by task and subtask for the project lifecycle of the BBMS.

5.2.2 Schedule Allocation

In figure 4, seen below, the entirety of the project schedule can be seen in a Gantt chart view. This view shows the dependency mapping between tasks, and isolates important milestones as called out by a diamond representation in the chart. Specifically, the entire schedule shows both project-related, development-related and delivery tasks as appropriate over the entire course of the project, which is approximately 18 weeks in duration. Due to the nature of the waterfall methodology in use (see section 6.1), changes in schedule to external forces, changes in requirements, or technical slips during the development cycle can lead to a range of time slips to the right, pushing the project past its agreed and intended delivery date of January.

ID	Task Name	Duration	Start	Finish
1	Form Teams	3 days	Fri 8/8/14	Tue 8/12/14
2	Initial Customer Engagement	1 day	Wed 8/13/14	Wed 8/13/14
3	Identify Customer Needs and High-Level Requirements	5 days	Thu 8/14/14	Wed 8/20/14
4	Develop Milestones	2 days	Thu 8/21/14	Fri 8/22/14
5	Define Budget	1 day	Mon 8/25/14	Mon 8/25/14
6	Develop Requirements	5 days	Tue 8/26/14	Mon 9/1/14
7	Deliver and Discuss Software Requirements Specificati	1 day	Tue 9/2/14	Tue 9/2/14
8	Software Design Review	7 days	Wed 9/3/14	Thu 9/11/14
9	Create Software Design Document	3 days	Fri 9/12/14	Tue 9/16/14
10	Deliver and Discuss Software Design Document	1 day	Wed 9/17/14	Wed 9/17/14
11	Create Software Testing Specification	7 days	Thu 9/18/14	Fri 9/26/14
12	Deliver and Discuss Software Testing Specification	1 day	Mon 9/29/14	Mon 9/29/14
13	Build Calendar Lookup Services	3 days	Tue 9/30/14	Thu 10/2/14
14	Build Calendar Lookup UI	5 days	Fri 10/3/14	Thu 10/9/14
15	Build User Information Services	3 days	Fri 10/10/14	Tue 10/14/14
16	Build User Information UI	5 days	Wed 10/15/1	Tue 10/21/14
17	Produce Initial User Documentation	2 days	Wed 10/22/1	Thu 10/23/14
18	Build and Package Alpha (v0.1)	1 day	Fri 10/24/14	Fri 10/24/14
19	Release 0.1 (Alpha)	0 days	Fri 10/24/14	Fri 10/24/14
20	Build Room Reservation Domain Object	2 days	Mon 10/27/1	Tue 10/28/14
21	Build Room Reservation Services	3 days	Wed 10/29/1	Fri 10/31/14
22	Build Room Reservation UI	5 days	Mon 11/3/14	Fri 11/7/14
23	Build Payment Domain Object	2 days	Mon 11/10/1	Tue 11/11/14
24	Build Payment Services	3 days	Wed 11/12/1	Fri 11/14/14
25	Build Payment UI	5 days	Mon 11/17/1	Fri 11/21/14
26	Build Guest Reservation Domain Object	2 days	Mon 11/24/1	Tue 11/25/14
27	Build Guest Reservation Services	3 days	Wed 11/26/1	Fri 11/28/14
28	Build Guest Reservation UI	5 days	Mon 12/1/14	Fri 12/5/14
29	Produce Extended Functionality User Documentation	2 days	Mon 12/8/14	Tue 12/9/14
30	Build and Package Beta (v0.5)	1 day	Wed 12/10/1	Wed 12/10/1
31	Release 0.5 (Beta)	0 days	Wed 12/10/1	Wed 12/10/1
32	Mid-Point Customer Engagement	1 day	Thu 12/11/14	Thu 12/11/14
33	Mid-Point User Acceptance Engagement	2 days	Fri 12/12/14	Mon 12/15/1
34	Build Reservation Scanning Services	3 days	Tue 12/16/14	Thu 12/18/14
35	Build Database Connectivity Services	5 days	Fri 12/19/14	Thu 12/25/14
36	Produce Final User Documentation	2 days	Fri 12/26/14	Mon 12/29/1
37	Build and Package Final (v1.0)	1 day	Tue 12/30/14	Tue 12/30/14
38	Release 1.0 (Final)	0 days	Tue 12/30/14	Tue 12/30/14
39	Final Customer Engagement	3 days	Wed 12/31/1	Fri 1/2/15
40	Final User Acceptance Engagement	2 days	Mon 1/5/15	Tue 1/6/15
41	Delivery of Product (BBMS v1.0 and Documentation)	0 days	Tue 1/6/15	Tue 1/6/15

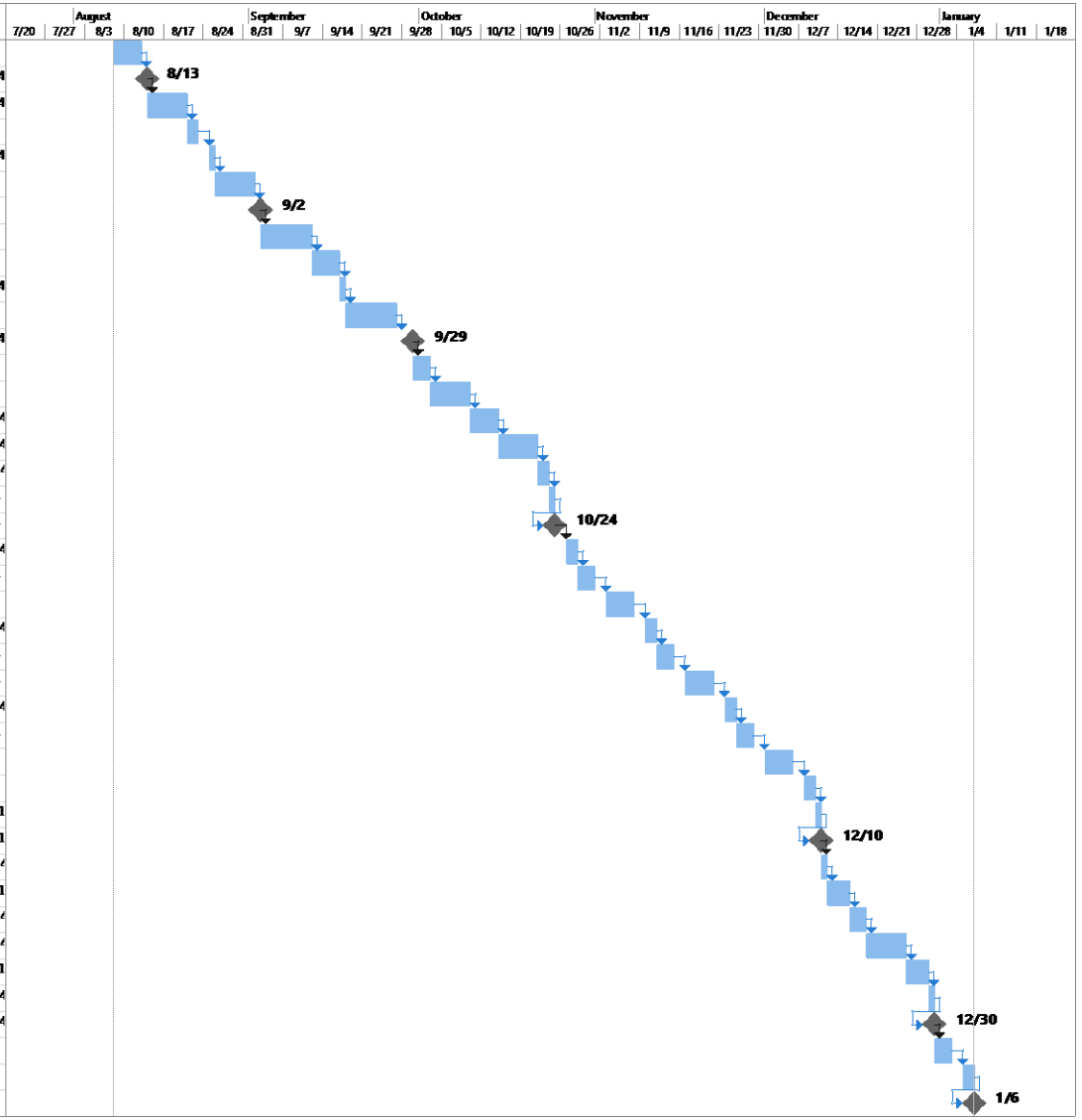

Figure 4. Gantt chart for Project Schedule

5.2.3 Resource Allocation

To correctly assess resource allocation for the BBMS project, an initial understanding of the complexity needs to be addressed regarding the overall design. To accommodate this, and subsequently associate a staff size estimate to the project, an overall analysis needs to be conducted using the Constructive Cost Model (COCOMO) method.

Since the BBMS consists of software components designed to approximate modeling of a mixture of payroll and inventory systems, the following analysis model of the project can be used via the COCOMO method:

COCOMO Mode	**Project Type**	**Team Size**	**Development Environment**	**Constraints and Deadlines**
Organic	Payroll, Inventory	Small	Stable (in-house)	Loose

As determined by the model, the use of the 'Organic' COCOMO mode can be determined for further analysis. This determination provides an effort equation in the form of:

$$Effort = a * (size) \wedge b$$

Where under the 'Organic' COCOMO mode, $a = 2.4$ and $b = 1.05$. A preliminary estimation of the source lines of code (*SLOC*) in thousands of units (*KLOC*) is determined to be 4.3, or ~4300 lines of code. This equation yields the following calculation for an estimation of human effort:

$$Effort = 2.4 * (4.3) \wedge 1.05 = \underline{11.10}$$

With an approximation of human effort, the project must now estimate project time in staff months. The use of the COCOMO method for the 'Organic' mode provides the equation as:

$$TDEV = 2.5 * (Effort) \wedge 0.38$$

With a calculated estimate of human effort, the calculated equation becomes:

$$TDEV = 2.5 * (11.10) \wedge 0.38 = \underline{6.24 \text{ months}}$$

In conclusion, an average staff size is calculated based on the ratio of estimated human effort versus estimated project staff time in months through the following equation:

$$Average\ Staff\ Size = Effort / TDEV$$

This equation yields the final calculation:

$$Average\ Staff\ Size = 11.10 / 6.24 = \underline{1.78}, \text{ or two full time developers}$$

The final calculation of two full time developers, in addition to staff providing customer liaison and project oversight functions, as well as user interface design, matches precisely with the mechanics required for a full implementation of the BBMS software. In addition the approximation of 6.24 months for development

is roughly equivalent to the calculated dependency tree of tasks seen via the project schedule in Figure 4, as well as the association of tasks in the Work Breakdown Schedule in Figure 3.

5.4 Risk Management Plan

Risk management for the BBMS is focused on the proper software configuration of databases and the ability for developers to prioritize features based on the customer requirement set. Specifically, the most critical risk is identified as the BBMS software not being able to accommodate for RDMBS implementations at the customer site. This is highlighted in figure 1.

Risk Priority	Description	Impact	Probability	Weighted Impact
1	Customer wants connection to unsupported database software.	$1000.00	0.5	$500
2	Customer wants feature not provided in requirements set	$500	0.8	$400
3	Customer is unable to load software	$3500	0.1	$350

Figure 4. Prioritized Risk Management Matrix

In order to accommodate for, and mitigate against the risk of database configuration and implementation mismatch (see Risk Priority 1, Figure 2), the development team must be able to provide the following:

1. Specific documentation stating what database software is supported by the BBMS.
2. Generate and apply specific tests against supported RDMBS.
3. Provide configuration examples for the supported RDBMS software.
4. Work with the customer, and provide support, for RDBMS software that is out-of-scope for integration.
5. Generate scenarios in documentation that explain ‘like’ configurations of databases. An example of this would be the use of MariaDB, which is similar in implementation to MySQL.

By enforcing a rigid set of examples and tests against the supported RDBMS software, working with the customer to ensure that these match specific requirements for development, and providing helpful support when the issue may arise, much of this risk is mitigated, providing a specific path on how to compensate for the risk.

6. Technical Process Plans

This section addresses the process and mechanics by which the BBMS development team will plan for, perform analysis of, construct, and test the source code based on customer requirements.

6.1 Process Model

The methodology to be used by the BBMS development team shall be constructed in a waterfall fashion for the purpose of program management. Internally, the development team may organize and develop within the assigned periods for deliverables in a pseudo-agile fashion, building epics, stories and tasks in conjunction with the overall planning strategy for the team. This model is heavily influenced by the Department of Defense model for government acquisition planning with lean development phases. This

model helps to achieve larger scale planning for acquisition and management, but provide more flexibility to the development teams tasked with implementation.

The process in place accommodates for a start (kickoff) activity based on project assessment and business analysis with and without the customer present. This provides for the development of both sub-processes and requirements which are used to generate milestones over the course of the project, and to assign responsibility of development tasks throughout the team. The conclusion of the project, also listed as a milestone, serves as the final delivery of the project after multiple cycles of customer engagement and user acceptance testing.

6.2 Methods, Tools, and Techniques

The BBMS development team, following the waterfall methodology for program management in section 6.1, will also require a standard process by which development, testing, integration and documentation is performed. The following standards are to be implemented and adhered to throughout the lifecycle of the project:

1. Language – Java and associated libraries
2. Database Implementations – MySQL, PostgreSQL, and Oracle
3. Source Code Repository – Git
4. Documentation – Microsoft Word 2013 and Adobe Acrobat
5. Testing – Junit
6. Build and Deploy – Maven

All source code is to be written using spaces versus tabs, and committed only upon successful completion of all relevant unit and integration tests. Any builds not conforming to this standard are to be rolled back to the prior Git commit hash, fixed, verified with automated testing, and pushed forward to the repository. This commit shall also include additional information specifying why the error was made, what was done to correct the action, and actions to be taken in the future to prevent it from occurring again.

Each build associated with a milestone in the project schedule (alpha, beta, final) will be built using automated build processes in Maven, and supplied to the customer liaison for discussion, demonstration and user acceptance testing. Failure to provide a timely build for the customer liaison will result in an internal review by all parties to address any issues leading up to the issue.