Team Seven Project Management Plan
Commerce Bank Productivity Analysis System

Eric Sundquist, Jeremy Szyba, Shane Taylor, Charlie Thompson, Matt Yale, Jasmin Zehic

October 7, 2015

Revision History

Date	Name	Description
10/1/2015	M. Yale	First commit
10/1/2015	Eric Sundquist	Updated formatting to match other documents, restructured to match outlines given in examples
10/6/2015	Eric Sundquist	Startup Plan, Work Plan
10/7/2015	M. Yale	

Overview

Purpose and Scope

The Commerce Bank Productivity Analysis System exists to help management track and measure goals. PAS is responsible for keeping a history of benchmarks and calculating scores based on whether certain measurements are meeting or exceeding their goals. With the improvements made by our team, PAS will have a user-friendly dashboard which allows for customizable widgets that show vital information relating to measurements and goals at a glance.

Our scope is the dashboard only at this time. If our scope changes it will have to go through proper change request procedures. Exceptions to this can occur in cases where the system code provided needs to be changed in order to accomplish our goals on the dashboard. If bugs are found in the existing code base and they have an easy fix, they will be fixed at the discretion of the team.

Some features we hope to implement include: unlimited widgets, customizable widgets, “storage boxes” for widgets that will be needed later, but are not currently needed on the dashboard, “stackable mode,” and more. As the project progresses and feedback from the client is gained, our direction will change and some of these features might not be implemented. We aim to provide the highest priority features to the client before moving on to others.

Goals and Objectives

Our goal is to deliver an intuitive GUI system for the Commerce Bank PAS system dashboard. The learning curve for new users should be under 10 minutes. The user should not (or rarely) get frustrated with the product.

The dashboard should incorporate useful features for the user, that are customizable, and are saved between user sessions.

Project Deliverables

September 11, 2015	Project charter
September 16, 2015	Product backlog
September 16, 2015	Use cases
September 21, 2015	Requirements document
September 23, 2015	Technical prototype
September 28, 2015	Iteration 1 begins
October 7, 2015	Project plan
October 11, 2015	Iteration 1 ends
October 12, 2015	Iteration 2 begins
October 12, 2015	5 minutes status report
October 14, 2015	Customer approved UI prototype

October 15, 2015	Demonstration
October 25, 2015	Iteration 2 ends
October 26, 2015	Iteration 3 begins
October 28, 2015	Architecture document
November 2, 2015	Mid-semester review with customer
November 8, 2015	Iteration 3 ends
November 9, 2015	Iteration 4 begins
November 29, 2015	Iteration 4 ends
November 30, 2015	Iteration 5 begins
December 2, 2015	User guide and system documentation
December 7, 20157	Project completed
December 11, 2015	Final presentation at Commerce Bank

Assumptions and Constraints

The project sponsor expects the project to be deployable in their environment. Our code must be integrateable into the customer's existing code. We must ensure the project will build and run within Visual Studio 2013 and .net 4.5.

It is assumed the customer will provide a thorough explanation of the problem domain and give clear guidelines on acceptable solutions. It is assumed that the customer will remain in contact with the team throughout the semester. We assume the customer will be able to provide feedback on seeing the current stages of the product throughout the semester.

Schedule and Budget Summary

9/11/15 - 9/27/15	Requirements gathering, product design, project planning, technical prototype
9/28/15 - 10/11/15	Iteration 1, first release of workable product
10/12/15 - 10/25/15	Iteration 2, status report, demonstration, customer-approved UI prototype
10/26/15 - 11/08/15	Iteration 3, mid-semester customer review, further product features and fixes
11/09/15 - 11/29/15	Iteration 4, further product features and fixes
11/30/15 - 12/11/15	Iteration 5, project completion, user documentation, customer presentation

Each member is expected to work on the project an average of six hours weekly, but on any given week will commit to working no less than three hours. We can therefore predict that, over the fifteen weeks of the project, our budget will be about 540 hours commited to the project.

Success Criteria

The project will be considered a success if the team produces:

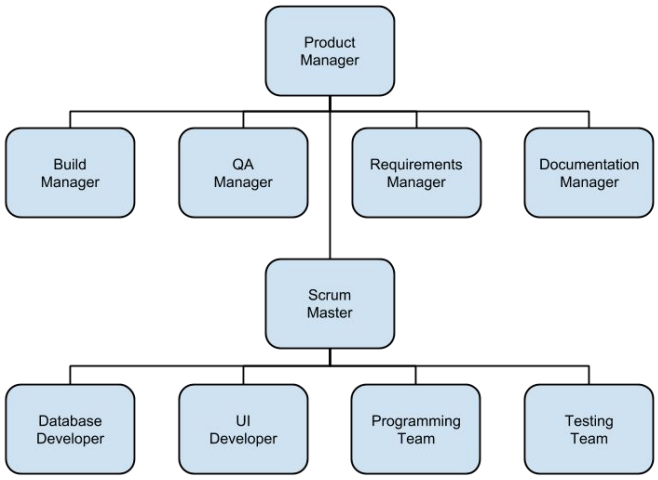
- A working UI that takes under 10 minutes to learn
- A customizable dashboard that persists its state between sessions
- Customizable widgets
- Ways to store widgets that might be needed later

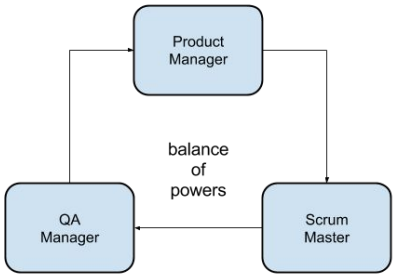
Definitions

Actor	a user or other software system that receives value from a use case
Customer	the person or organization that will pay for the product, namely, Commerce Bank
May	indication of an optional requirement, allows for an option rather than a requirement
PAS	Productivity Analysis System, the product being developed
Product	the PAS being described here; the software system specified in this document
Project	activities that will lead to the production of the product described here
Shall	indication of a mandatory requirements, also “must” and “will”
Should	indication of a desired requirements, though not mandatory
Use case	a goal-oriented interaction between the system and an actor
User	the person or persons who will interact with the product, namely, Commerce Bank managers

Startup Plan

Team Organization





Product Manager - Matt Yale. Product Manager oversees entire project and all team leaders. Decides features to implement each iteration. Responsible for team submissions. Communicates with customer. Can organizationally override Scrum Master.

Build Manager - Shane Taylor. Build Manager oversees source control. Sets rules for commits. Ensures code commits do not break status of overall project. Reports to Product Manager.

QA Manager - Eric Sundquist. QA Manager responsible to ensure code is easy to understand, easy to maintain, and robust. Quality assurance also responsible to ensure the code works as the customer expects it to. Reports to Product Manager and can also organizationally override Product Manager.

Requirements Manager - Eric Sundquist. Requirements Manager works with Product Manager and Customer to define the requirements of the project. Assists Scrum Master to prioritize features. Reports to Product Manager.

Documentation Manager - Eric Sundquist. Documentation manager responsible to provide documentation of product and project to team and customer. Has say over final versions of documents. Reports to Product Manager.

Scrum Master - Jeremy Szyba. Scrum Master oversees programming team and leads scrum meetings.Organizes team meetings. Assigns tasks to programming team. Reports to Product Manager, and can organizationally override QA Manager.

Database Developer - Shane Taylor. Database Developer coordinates product connection to a database and assist programmers in database related areas. Reports to Scrum Master.

UI Developer - Jeremy Szyba. UI Developer coordinates product's UI. has final say in product's final UI and features it implements. Assists programmers in UI related areas. Reports to Scrum Master.

Programming Team - Charlie Thompson, Shane Taylor, Jasmin Zehic. Programming Team writes code and test cases. Responsible to provide working features at every iteration. Reports to Scrum Master.

Testing Team - Charlie Thompson, Jasmin Zehic. Testing team writes test cases for code and ensures that code performs as expected in versatile situations. Testing team responsible for providing test cases for each controller to customer with final product. Reports to Scrum Master.

Project Communications

Iteration planning meetings happen every two weeks at the beginning of the iteration. They are in-person meetings including the entire team. Their main purpose is to decide which features get implemented in the next iteration and assign responsibilities for specific features.

Informal scrums happen after class. They are in-person, impromptu meetings including the entire team. Their main purpose is to discuss progress being made and any blockages.

Iteration reviews happen every two weeks following an iteration. They are in-person meetings including the entire team. Their main purpose is to discuss what worked procedurally and what needs to be changed to allow the next iteration to succeed.

Email is used frequently. It may include the entire team or the individuals affected by the subject of the email. It may also be between the team leader and the customer. Their purpose is to discuss progress being made and any blockages.

Status reviews happen about every month in class. They include the entire team and the entire class. Their purpose is to inform the class and instructor of the large-scale progress being made on the product.

There will be one mid-semester customer presentation and one end-semester customer presentation. The presentations include the entire team and the customer. Their purpose is to instruct the customer in the progress made on the product, its use, and to elicit feedback.

Technical Process

The project will follow an agile programming process. Iterations will be scheduled for every two weeks, with usable features delivered at each iteration. Feedback is used to improve the product and add requested features.

Tools

The product will be built inside Visual Studio using the .NET framework. Languages include C#, JavaScript, SQL, HTML, CSS, and XML. GitHub will be used for source control. TeamCity will be used for integration support.

Work Plan

Release Plan

The following features are scheduled to be released.

- Widgets are able to be added and removed from the dashboard
- Widgets are able to be resized and moved around
- Widgets will display customizable representations of the data in a database
- Widgets will display best and worst performing goals
- The dashboard will save its configuration between uses

Iteration plans

Iteration 1:

- Widgets are able to be added and removed from the dashboard
    - Provide abstract widget class
    - Widgets can be added by clicking a “plus” icon
    - A “minus” icon appears when there are over 0 widgets
    - Widget clears when clicked in remove mode
- Widgets will display customizable representations of the data in a database
    - Widget will reflect some pre-determined data in a database

Iteration 2:

- Widgets are able to be resized and moved around
    - Widgets can be dragged and dropped on dashboard
    - Widget can be easily resized
- Widgets will display customizable representations of the data in a database
    - Users can select specific goals to track measurements
    - Users can track if a goal is over- or under-performing
    - Widget displays data in a chart

Iterations 3+:

- TBD

Control Plan

Monitoring and Control

Base-lined documents must adhere to the following rules.

- All base-lined documents are stored in a shared Google Drive folder
- All base-lined documents include near the beginning a history section, where users record the person, date, and nature of each change
- All changes to base-lined documents need to be approved by the Product Manager, and Documentation Manager. Changes to requirements must additionally be approved by the Requirements Manager.
- Base-lined documents are to be reviewed by all group members after creation before becoming base-lined. Any concerns must be addressed at this time.

Metrics Collection

The project will use no formal metrics collection. The customer has requested programmers to track how much time they spend researching and learning technologies versus how much time they spend writing code. These estimates will be provided at the end of the semester upon product completion. No other metrics will be collected.

Supporting Process Plans

Risk Management Plan

Rank	Risk	Prob x Mag = R.E.	Response
1	Difficulties adapting technology	3 x 2 = 6	Buy information: create prototype of needed technology
2	Unable to complete in given time	3 x 2 = 6	Avoid: use iterative and incremental programming techniques so there is a working product after first iteration
3	Team members unable to provide estimated contributions	2 x 2 = 4	Avoid: implement regular meetings to ensure everyone is able to contribute to project
4	Customer changes expectations for project	1 x 3 = 3	Mitigate: use modularized code which can be reused and easily swapped out
5	Bugs in code customer provided impede progress	2 x 1 = 2	Transfer: allow customer to take responsibility for the bugs they introduced
6	Team underestimates amount of work	1 x 2 = 2	Buy information: communicate with customer regularly to ensure project scope is well-defined
7	Requirements poorly understood	1 x 1 = 1	Buy information: communicate with customer regularly to ensure project scope is well-defined
8	Difficulty communicating with customer when needed	1 x 1 = 1	Transfer: allow customer to take responsibility if they are not involved in project