**Project Plan**

**for the**

**SIUE Department of Computer Science**

**CS425 / CS499 Senior Project**

**Software Design and Implementation Courses**

**by**

**Zach Benchley, Matt Lievens, Logan Maughan, Brian Olsen**

**of**

****

**Learning Outcomes-Based Assessment Database Team**

Revision 1.1

As Of: September 28, 2013

LOBA-PP

Change Log:

|  |  |
| --- | --- |
| Revision | Change Note(s) |
|  |  |
| 1.0 | * Initial Release |
| 1.1 | * Updated Schedule Dates * Updated Table of Contents and List of Figures * Updated Table of Milestones and Tasks * Removed Connection With Blackboard From Architecture Section |

Reviewed and Approved By:

Name Signature Date

Table of Contents

[1 Project Estimation 3](#_Toc373958741)

[1.1 Architecture 3](#_Toc373958742)

[1.2 Time Estimation. 3](#_Toc373958743)

[2 Resources 6](#_Toc373958744)

[3 Scheduling 8](#_Toc373958745)

[4 Communication 10](#_Toc373958746)

[4.1 Configuration Management Tool 10](#_Toc373958747)

[4.2 Online File System 10](#_Toc373958748)

[4.3 Integrated Development Environment 10](#_Toc373958749)

[4.4 Documentation 11](#_Toc373958750)

[4.5 Messaging 11](#_Toc373958751)

[5 Quality Assurance 11](#_Toc373958752)

[5.1 Testing Plan 11](#_Toc373958753)

[5.2 Risk Table 13](#_Toc373958754)

[6 Exit Strategies 13](#_Toc373958755)

[6.1 425 Exit Strategy 13](#_Toc373958756)

[6.2 499 Exit Strategy 14](#_Toc373958757)

[7 References 14](#_Toc373958758)

# List of Figures

Figure 1 - Test First Development Flow chart 12

Figure 2 - Acceptance/Developer TDD Flow Chart 13

# 1 Project Estimation

Below is the project estimation, which includes a summary of our systems architecture and a summary of our project’s time estimation.

## 1.1 Architecture

When developing the LearnOBA software system there are many requirements that are going to need to be met. The team developed a list of these requirements and made a high-level architecture of the system. This architecture contains three main sub-systems: the LearnOBA database, the LearnOBA graphical user interface (GUI), and the SIUE Blackboard system. The SIUE Blackboard system has already been developed and the team will just have to facilitate information to and from Blackboard. There will be no modifying of the Blackboard system. The LearnOBA GUI is going to be able to export data from the database into a format that Blackboard can read. All human interactions will go through the GUI. Upon the proper command given by the user, the GUI can store and update things in the LearnOBA database. A diagram of the architectural flow of the system can be observed in the LOBA-PP-CLASS, LOBA-PP-SEQUENCE, and LOBA-PP-DATABASE documents.

## 1.2 Time Estimation.

With the architecture being laid out the team was now able to develop a list of major milestones that must be completed to ensure a successful software system. The list of these milestones and the estimated time to complete each milestone is flushed out in the table below.

|  |  |  |
| --- | --- | --- |
| **StoryID** | **Tasks** | **Estimated** |
|  | **CS 425: Implementing, Testing, and Demonstrating** |  |
|  | **Designing Database** |  |
| 14 | Create ER Diagrams(Logan) | 7.5 |
| 14 | Create Class Diagrams(Logan) | 7.5 |
| 16 | Create Table Relation Diagram(Matt) | 7.5 |
| 16 | Research integrateing SQlite with JavaFX (Logan) | 7.5 |
|  | **Implementing Database** |  |
| 19 | Research Creating Database in SQlite (Matt) | 4 |
| 3 | Create Class Table SQL (Matt) | 2 |
| 4 | Create Student Table SQL (Matt) | 2 |
| 10 | Create Skill Table SQL (Matt) | 2 |
| 8 | Create Task Table SQL (Matt) | 2 |
| 12 | Create Score Table SQL (Matt) | 2 |
| 1 | Create User Table(Matt) | 2 |
| 18 | Create Sqlite Database using code from .sql files(Matt) | 4 |
|  | **Implementing Database Queries (SQL)** |  |
| 15 | Create Select Query Test Class (Brian) | 4 |
| 22 | Create Update Query Test Class (Brian) | 4 |
| 3 | Create Insert Query Test Class (Brian) | 4 |
| 4 | Create Delete Query Test Class (Brian) | 4 |
| 10 | Export Backup Database Capability(Matt) | 4 |
| 8 | Create Select Query Class (Matt) | 4 |
| 12 | Create Update Query Class (Matt) | 4 |
| 1 | Create Insert Query Class (Matt) | 4 |
| 22 | Create Insert Query Class (Logan) | 4 |
| 13 | Create Delete Query Class (Logan) | 4 |
| 17 | Create Database Façade Class (Logan) | 5 |
| 16 | Create Abstract Query Class (Logan) | 5 |
|  | **Designing GUIs (Class & Prototype)** |  |
| 1 | Login Screen Design (LoginGUI) | 4 |
| 19 | Welcome Screen Design (WelcomeGUI) | 4 |
| 2 | Create Class Design (CreateClass) | 2 |
| 18 | LearnOBA Main Driver Design (LearnOBA Main & MainTabGUI) | 4 |
| 3 | Selecting Classes Design (SelectClass) | 4 |
| 4 | Adding Students to Class Design (SelectClass) | 3 |
| 5 | Deleting Students from Class Design (SelectClass) | 3 |
| 6 | Adding Assignments to Class Design (SelectClass) | 3 |
| 7 | Removing Assignments from Classes Design (SelectClass) | 3 |
| 8 | Adding Tasks to Assignment (EditAssignment) | 3 |
| 9 | Removing Tasks from Assignment (EditAssignment) | 3 |
| 10 | Adding Skills to a Task (EditSkills) | 3 |
| 11 | Removing Skills from a Task (EditSkills) | 3 |
| 12 | Entering Student Scores (EnterScores) | 3 |
|  | **Learning Curves** |  |
| 21 | Learning JavaFX (Zach) | 5 |
| 21 | Learning JavaFX (Brian) | 7 |
| 21 | Learning JDBC and SQLite (Matt) | 5 |
| 21 | Learning JDBC and SQLite (Logan) | 5 |
| 21 | Learning JDBC and SQLite (Brian) | 6 |
| 19 | Multi-Platform Build (Apache Ant) (Brian) | 6 |
| 19 | (Set up snags with JavaFX)(Brian& Zach) | 3 |
|  | CS 425 TOTALS | 182 |
|  | **CS 499: Implementing, Testing, and Demonstrating** |  |
|  | **Implementing GUIs** |  |
| 1 | Login Screen Implementation (LoginMVC) | 5 |
| 19 | Welcome Screen Screen Implementation (WelcomeMVC) | 5 |
| 2 | Adding Classes Implementation (CreateClassMVC) | 6 |
| 19 | LearnOBA Main Implementation (LearnOBA Main & MainTabMVC) | 6 |
| 3 | Selecting Classes Implementation (SelectClassMVC) | 6 |
| 4 | Adding Students to Class Implementation (SelectClassMVC) | 6 |
| 5 | Deleting Students from Class Implementation (SelectClassMVC) | 6 |
| 6 | Adding Assignments to Class Implementation (SelectClassMVC) | 6 |
| 7 | Removing Assignments from Classes Implementation (SelectClassMVC) | 6 |
| 8 | Adding Tasks to Assignment Implementation (EditAssignmentMVC) | 6 |
| 9 | Removing Tasks from Assignment Implementation (EditAssignmentMVC) | 6 |
| 10 | Adding Skills to a Task Implementation (EditSkillsMVC) | 6 |
| 11 | Removing Skills from a Task Implementation (EditSkillsMVC) | 6 |
|  | **Testing of GUIs** |  |
| 11 | Removing Skills from a Task Unit Testing (EditSkillsMVC) | 3 |
| 1 | Login Screen Unit Testing (LoginMVC) | 3 |
| 19 | Welcome Screen Unit Testing (WelcomeMVC) | 1 |
| 2 | Adding Classes Unit Testing (CreateClassMVC) | 3 |
| 3 | Selecting Classes Unit Testing (SelectClassMVC) | 3 |
| 9 | Removing Tasks from Assignment Unit Testing (EditAssignmentMVC) | 3 |
| 10 | Adding Skills to a Task Unit Testing (EditSkillsMVC) | 3 |
| 8 | Adding Tasks to Assignment Unit Testing (EditAssignmentMVC) | 3 |
| 7 | Removing Assignments from Classes Unit Testing (SelectClassMVC) | 3 |
| 6 | Adding Assignments to Class Unit Testing (SelectClassMVC) | 3 |
| 5 | Deleting Students from Class Unit Testing (SelectClassMVC) | 3 |
| 4 | Adding Students to Class Unit Testing (SelectClassMVC) | 3 |
| 19 | LearnOBA Main Testing (LearnOBA Main & MainTabMVC) | 3 |
| 13 | Integration Testing | 40 |
| 18 | System Testing | 20 |
| 12 | Method for Creating Reports | 5 |
| 15 | **Getting application to run on Mac** | 10 |
| 17 | **Getting application to run on Windows** | 10 |
| 22 | **Getting application to run on Linux** | 10 |
| 11, 9 | \*\*Other methods that aid in full implementation | 20 |
| 5 | Demonstration and Delivery | 20 |
|  | Total: | 440 |

\*\*Note: Actual times for tasks that have not yet been completed are copied over from estimated time. This is so that we can ­­­have an 'actual total' to compare our estimated total to.

Initially it appeared that the project would take a total of 470 person hours. This means that each person on the team would have put in around 8.5 person hours a week over the course of seven sprints. The team has came to the conclusion that we are able and willing to put in this much time each week which will allow us to be able to complete a very successful software system. The times given above are subject to change and will be updated when necessary.

During sprint 2 the team made a few changes to the initial estimates including: reallocating time to include time for learning the new software, and removing the time allocated for integrating LearnOBA to Blackboard. After applying these changes the total estimated person-hours down to 440 hours.

As of the end of sprint 2 the team now estimate that there are 271 person-hours remaining in this project. Based off the current projected person-hours, it is estimated that the each team member will now need to work 6.7 person-hours per week to complete this project.

# 2 Resources

Below are the resources that the team will be using throughout the development and implementation of this software product.

1. Dr. Tom Foster (Product Owner) – The team will be consulting with Dr. Foster throughout the entire process. He is the client, therefore his opinion and viewpoint on certain issues regarding the software are needed.
2. Dr. Gary Mayer (Oversight) – The team will be consulting with Dr. Mayer throughout the entire process. If any questions arise that cannot be answered by Dr. Foster or that are about the SAGE process specifically, he will be able to answer them.
3. The Team –
   1. Zach Benchley (Scrum Master): The Scrum Master is responsible for representing the team to Oversight. Other responsibilities include – enforcing the SAGE Process, keeping the team on task and working together, and maintaining both a Product Backlog and Product Burndown Chart.
   2. Matt Lievens (Database Lead): The Database Lead is responsible for overseeing the development of the database. Other responsibilities include – maintaining all diagrams and models developed in the creation of the database and making sure all data is kept up to date.
   3. Brian Olsen (Quality Controller): The Quality Controller is responsible for leading the development and implementation of the team’s quality control strategy. Other responsibilities include – Maintaining the risk tables associated with the product and keeping records of test results.
   4. Logan Maughan (Owner Proxy): The Owner Proxy is responsible for keeping an open channel of communication between the Product Owner and the team. Other responsibilities include – Having an extensive knowledge of the client domain, maintaining the requirements list, and guiding the Product Owner through the SAGE Process.
4. ITS – The team will be contacting ITS in order to gain special access so that the program can communicate with Blackboard.
5. Computer – Each team member will need a computer that can be used for this project. More specifically, the computer must be able to create and update documents for the SAGE Process Specification and write and compile code using Java and MySQL.
6. Mac, Linux, and Windows – One of the requirements states that the final product must be able to run on Mac, Linux, and Windows. During each and every testing phase, the tests will be run on each of these operating systems. One of the computers from the Senior Project lab will be used for Mac OS, and a team member’s personal computer will be used for Linux and Windows.
7. Software (Eclipse, JavaFX, and SQLite) – The project will be written in JavaFX and SQLite using the Eclipse IDE. Each team member should have all three of these readily available in order to work on the project.
8. Time – As stated in the estimation section of the Project Plan, it will take 470 person hours over the next two semesters to complete this project.

# 3 Scheduling

Below is the schedule that the team will be using throughout the development and implementation of this software product. Unless otherwise specified, all sprints will be two weeks in length.

Individual Workdays:

|  |  |
| --- | --- |
| Zach Benchley | Tuesday: 6:00 pm – 8:00 pm  Thursday:6:00 pm – 8:00 pm  Friday: 12:00 pm – 4:00 pm |
| Matt Lievens | Friday: 6:00 pm – 9:00 pm  Saturday: 2:00 pm – 5:00 pm  Sunday: 10:00 am – 12:00 pm |
| Brian Olsen | Thursday: 8:00 am – 12:00 pm  Friday: 6:00 pm – 10:00 pm  Saturday: 4:00 pm – 8:00 pm |
| Logan Maughan | Monday: 8:00 pm – 10:00 pm  Wednesday: 8:00 pm – 10:00 pm  Saturday: 4:00 pm – 7:00 pm |

|  |  |  |
| --- | --- | --- |
| 2013 | | |
| **September** | **29** | Install Needed Software (Eclipse, Java, JavaFX, SQLite) |
| **October** | **1** | Project Plan Draft Complete |
|  | **6** | Understand Java, JavaFX, and SQLite. Know how to use Eclipse. |
|  | **8** | Project Plan Due |
|  | **10** | Sprint 1 Starts (Database and GUI Prototypes) |
|  | **15** | In Person Daily Scrum Meeting |
|  | **15** | Project Specification Presentation |
|  | **17** | In Person Daily Scrum Meeting |
|  | **17** | Project Plan Presentation |
|  | **22** | In Person Daily Scrum Meeting |
|  | **24** | In Person Daily Scrum Meeting |
|  | **24** | Sprint 1 Review and Retrospective Meeting with Dr. Foster |
|  | **31** | Status Report Due |
|  | **31** | Sprint 2 Starts (Finish Database and GUI Design) |
| **November** | **5** | Prototype/Design Presentation |
|  | **5** | In Person Daily Scrum Meeting |
|  | **7** | In Person Daily Scrum Meeting |
|  | **12** | In Person Daily Scrum Meeting |
|  | **14** | In Person Daily Scrum Meeting |
|  | **19** | Sprint 2 Review and Retrospective Meeting with Dr. Foster |
|  | **22** | Final Presentations |
|  | **25-29** | Thanksgiving Break - No Classes |
|  | **28-30** | Thanksgiving Holiday - University Closed |
| **December** | **5** | Post Mortem Due |
|  | **7-13** | Final Exams |
|  | **14-30** | Winter Break |

|  |  |  |
| --- | --- | --- |
| 2014 | | |
| **January** | **1-12** | Winter Break |
|  | **13** | First Day of Weekday and Evening Classes |
|  | **14** | Meet with Dr. Foster for Project Update |
|  | **16** | Sprint 3 Starts |
|  | **20** | Dr. Martin Luther King, Jr. Day - University Closed |
|  | **21** | In Person Daily Scrum Meeting |
|  | **23** | In Person Daily Scrum Meeting |
|  | **28** | In Person Daily Scrum Meeting |
|  | **30** | In Person Daily Scrum Meeting |
|  | **30** | Sprint 3 Review and Retrospective Meeting with Dr. Foster |
| **February** | **4** | Sprint 4 Starts |
|  | **6** | In Person Daily Scrum Meeting |
|  | **11** | In Person Daily Scrum Meeting |
|  | **13** | In Person Daily Scrum Meeting |
|  | **18** | In Person Daily Scrum Meeting |
|  | **18** | Sprint 4 Review and Retrospective Meeting with Dr. Foster |
|  | **20** | Sprint 5 Starts |
|  | **25** | In Person Daily Scrum Meeting |
|  | **27** | In Person Daily Scrum Meeting |
| **March** | **4** | In Person Daily Scrum Meeting |
|  | **6** | In Person Daily Scrum Meeting |
|  | **6** | Sprint 5 Review and Retrospective Meeting with Dr. Foster |
|  | **10-14** | Spring Break |
|  | **18** | Sprint 6 Starts |
|  | **20** | In Person Daily Scrum Meeting |
|  | **25** | In Person Daily Scrum Meeting |
|  | **27** | In Person Daily Scrum Meeting |
| **April** | **1** | In Person Daily Scrum Meeting |
|  | **1** | Sprint 6 Review and Retrospective Meeting with Dr. Foster |
|  | **3** | Sprint 7 Starts |
|  | **8** | In Person Daily Scrum Meeting |
|  | **10** | In Person Daily Scrum Meeting |
|  | **15** | In Person Daily Scrum Meeting |
|  | **17** | In Person Daily Scrum Meeting |
|  | **17** | Sprint 7 Review and Retrospective Meeting with Dr. Foster |
|  | **22** | Delivery |
|  | **25** | Post Mortem |
| **May** | **3-9** | Final Exams |

# 4 Communication

Below are the following tools used for communication between Client, Oversight, and the team throughout the development process.

## 4.1 Configuration Management Tool

Bitbucket by AtlassianBitbucket is an online repository that will allow the team to share, version and manage the code. The management can be done through console commands but due to the team’s lack of familiarity with the Bitbucket syntax, a Graphical User Interface provided by Atlassian will be used.

* 1. Graphical User Interface:
     1. Source Tree - A free PC application that allows the team to manage the software and gives helpful visuals.
     2. Bitbucket.org – the website provided is also a sufficient way to manage the software.

## 4.2 Online File System

Dropbox will be used to give the team the ability to share and store documents over the internet

1. Team Folder – the team folder serves as an online storage folder while making updates and changes amongst documents and software releases. This will not be visible to the client as to avoid confusion during the updating process.
2. Client Folder – the client folder is a “cleaned up” version of what is kept in the team folder. It will allow the team to show the client only what is needed to be seen.

## 4.3 Integrated Development Environment

To help manage syntax and similar folder hierarchies, the team has chosen to use Eclipse. Eclipse not only offers a vast array of organizational tools but a variety of plug-ins and tools that will help during development. It is also worth noting that Eclipse integrates smoothly with Java and specifically JavaFX.

## 4.4 Documentation

Meeting minutes and logs will be held on Microsoft Excel spreadsheets and Microsoft Word documents. Class Diagrams will be made using either Microsoft Visio or Dia software.

## 4.5 Messaging

For immediate calls and messages, the team will keep in contact through calling and texting via cellular telephone. On certain occurrences online mediums such as Facebook and Skype will be supplemented for messaging. For more official and non-urgent messages the team will use email as a means to fulfill message relay, as well as documentation.

# 5 Quality Assurance

## 5.1 Testing Plan

The team will execute testing throughout each sprint, and more specifically before any functional code is written. The general model executed is going to follow Test Driven Development.

1. Test First Development – This is the core idea of Test Driven Development where test code is written before any functional code. A popular approach is to write functional code in small steps until the original test is satisfied. The main rule to be followed is, “No Test? No Code!” Figure 1 models the steps of Test First Development.



Figure 1 - Test First Development Flow chart

1. Refactoring – Test driven development is a composition of test first development and refactoring. There are different types of refactoring based on the role of the team member. The type of refactoring ultimately determines what type of test driven development is being applied. Below are the following types of refactoring and correlating roles.
   * 1. Behavior Driven Development – There is no test code nor functional code written during this development. This step is handled and maintained by the Owner Proxy by gathering information and flushing out requirements. These requirements are then flushed out into Agile use cases that can be used to develop acceptance tests. This step is handled mainly in the Concept phase and is continued each preparation phase in each iteration.
     2. Acceptance Driven Development – In this stage the first test code is written while the functional code is optional (since it is expected to fail.) The test code produced is called the acceptance test and its primary goal is to check the behavior of a given module. The important thing to remember in this stage is to avoid testing the functionality of the code (i.e. the methodology and efficiency of the code). The acceptance test executes black box testing as it gives a set of inputs and expects a specific output. The tests can be implemented either as a JUnit test to test behavior of a module or an integration test to check interaction between 2 or more modules. This can be the responsibility of the Team or the Scrum Master.
     3. Developer Test Driven Development – This is the stage that is iterated along with an acceptance test. A developer test is written to check for specific functionality. This test is analogous to a white box test as it has knowledge of the inner-workings of the module. This test can be written to fit all the requirements at once or can be iterated one method at a time. These tests are specifically JUnit tests to test specific functionality of separate modules. It is likely that once all developer tests have passed that the acceptance test will pass as well. If not make sure to check everything has been tested thoroughly.

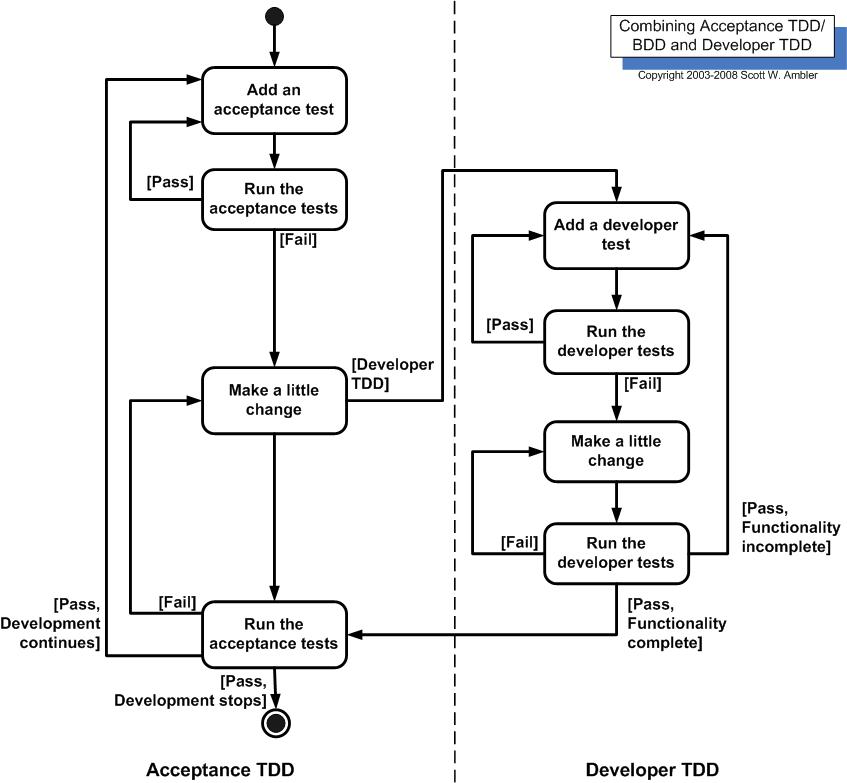


Figure 2 - Acceptance/Developer TDD Flow Chart

## 5.2 Risk Table

Risk table will be maintained in the LOBA-PP-RISK document.

# 6 Exit Strategies

## 6.1 425 Exit Strategy

The team’s exit strategy for CS 425 will be the creation of a Core Component along with the creation of a Major Component Prototype. The reason the team decided to follow this course is that the two strategies will complement each other well.

The Core Component will be the database that this project centers around. The team’s goal is to have the Database built and fully operational. The testing of this database will follow the test driven development standards set in the Quality Assurance section. Additional testing will be done using the user interface developed for the Major Component Prototype.

The Major Component Prototype will be LearnOBA’s user interface. The user interface that is developed through the Major Component Prototype will be used to demonstrate what LearnOBA will eventually look like. Along with demonstrating what LearnOBA will look like, the Prototype will also be used to test ways for the user interface to interact with the database. This user interface will also be used to test the Core Component that will be developed in CS 425.

## 6.2 499 Exit Strategy

The exit strategy for CS 499 details how the team will be developing, testing, and demonstrating LearnOBA. It also goes over the team’s plans for documenting and deploying LearnOBA for the Product Owner, and all future users.

As specified in the Quality Assurance section of the Project Plan, LearnOBA will be developed using Test Driven Development techniques. Due to this type of development most major tests will happen as the various areas of this project are developed.

Once an initial user interface is made the team would like to demonstrate LearnOBA to the Product Owner after every successful sprint. By demonstrating often the team will be able to obtain useful feedback from the Product Owner about the current changes, enabling the Product Owner and the team to remain on the same page throughout the development process.

Due to the limited number of users for LearnOBA, some of the demonstrations with the Product Owner will also be used as a pseudo usability study. These studies will enable the team to make LearnOBA easy to use for its users while still meeting all of the requirements for the software.

The team will also create documentation to help future LearnOBA users with any questions they have in regards to using the software.

Once development has ended and the documentation has been written, the team will deliver the finished version of LearnOBA to the Product Owner. LearnOBA will be able to be installed onto a computer from a file, located on a CD or a flash drive.

# 7 References

Ambler, Scott. "Introduction to Test Driven Development (TDD)." *Introduction to Test Driven Development (TDD)*. Ambysoft Inc., n.d. Web. 04 Sept. 2013. [http://www.agiledata.org/essays/tdd.html#TraditionalTesting](http://www.agiledata.org/essays/tdd.html)