McGill

**ECSE 321**
**Introduction to Software Engineering**

**Home Audio System**
**Deliverable 4**
**March 31, 2016**

**Kristina Pearkes 260 520 832**
**Aidan Piwowar 260 625 505**
**Kareem Halabi 260 616 162**
**Aurelie Pluche 260 622 575**
**Alexander Orzechowski 260 610 696**

**Release Pipeline Plan**

Integration Phase of all Platforms

For the integration phase of the release pipeline, all platforms are treated similarly. We utilize Git and Github to our advantage when updating and pushing code. We are able to see the code insertions and deletions in the code base through Github. This gives immediate feedback and allows for quick and efficient updating. The code is pushed onto a development branch (a separate entity from the master-branch) in order for the build phase to begin.

Build Phase

***Web:*** We will be using Travis CI to continuously integrate the changes we make and to trigger apache ant to build the various files for the web application and verify that all tests pass. The testing is done automatically through the ant script. Only once all the tests pass will the build files be created and can be deployed.

***Android:*** Travis CI uses Gradle is to compile all the classes and dependencies used for the android application every time the source code is updated on GitHub. Since the android application utilizes the desktop jar (which gets tested before being pushed), it does not need to be tested again when the android code is updated.

***Desktop:*** Travis CI is also used to continuously integrate the changes we make to the desktop application and to trigger apache ant to build and test out files. Travis CI automatically accesses the Git repository for the project and identifies changes within the code to generate a new build. All JUnit tests are run every time the build is executed to verify that the application runs correctly. Should the tests fail, Travis CI will give a notification and exit the build. If needed, the build will be created manually using apache ant directly, rather than using Travis CI as an intermediary.

Deployment Phase

***Web:*** The deployment phase for the Web application is pushing the application to the internet through a web hosting service once the build phase is complete. We use xampp Apache web hosting service to host our website. Once the website is online, the clients are able to access it. Once the build files have been created with the newest patch and are ready for release, the development branch will merge with the master branch as defined in the .travis.yml file. This will deploy the application as a GitHub release.

***Android:*** After a successful build and updates are ready for release, the development branch will be merged with the master branch. Travis will generate an apk file as a GitHub release that can be installed on a client's android device.

***Desktop:*** Once the build phase is complete, ant will have created a build file that can be used to create an executable jar. Once the updates are ready for release the development branch will be merged with the master branch. This will deploy the application as a GitHub release as defined in our .travis.yml file.

## Rationale to Support Design Choice

In the design of our Release Pipeline, we choose to use Travis CI as our integrating service to build and test the Home Audio System mainly because it's the service that the team members are most comfortable with. Additionally, Travis CI relies on ANT and Gradle for build tasks which are simple and easy to use. Moreover, we designed our release pipeline this way because we want our pipeline to achieve continuous delivery. In other words, if we need a bug fix or an exciting new feature, we want our team members to get instant feedback when they commit their updated code to the repository. Then, if what was added is of high quality, we want to get the product to the customer in a quick and efficient manner. Our design of the release pipeline meets this criteria for the development of all 3 applications.

**Updated Work Plan**

- Deliverable 1 Completed on February 22nd (High effort)
    - Add an album (2.1)
    - Add an artist (2.3)
    - Add a song to album (2.2)
    - Associate album to artist (3.1)
    - Functional and Non-Functional requirements
    - Domain Model
    - Use Case Diagram
    - Use Cases
    - Requirements-level sequence diagram for "Add Album" use case
    - Prototype of application on all three platforms for "Add Album" use case
- Deliverable 2 Completed on March 7th (Medium effort):
    - Add a room (1.2)
    - Description of architecture of proposed solution including a block diagram
    - Description of detailed design of proposed solution  including class diagram
    - Updated umple model, see below
- Deliverable 3 Completed on March 25th (High effort):
    - Create playlist(2.4)
    - Function in multiple rooms (1.3)
    - Organize music by album or artist (3.2)
    - Choose a volume level for a room (1.4)
    - Be able to mute a room(1.5)
    - Description of unit testing
    - Description of component testing
    - Description of system testing
    - Description of performance/stress testing
- Deliverable 4 by April 1st (Low effort)
    - Description of release pipeline.
    - Select an item to play (1.1)
    - Organize music by album or artist for php(3.2)
    - Finalize view for desktop application
- Deliverable 5 by April 11/12/14 (Medium effort)
    - Presentation about project.
    - Rehearse Presentation
    - Test final version of the application
- Deliverable 6 by April 15 (Medium-High effort)
    - Source code of full implementation on each supported platform