Jarvis Emulator  
Project Management Plan  
COP 4331, Fall 2015

**Modification History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Who** | **Comment** |
| v0.0 | 9/13/2015 | Robin Schiro | Created document |
| V1.0 | 9/15/2015 | Jimmy Lam | Added Project Team Organization |
| V2.0 | 9/15/2015 | Jimmy Lam | Added Technical progress metrics and plan for tracking, control and reporting of progress |
| V2.1 | 9/17/2015 | Manuel Gonzalez | Added Quality Assurance, Table of Work Packages and PERT Chart |

**Team Members:**

* Jimmy Lam
* Julian Rojas
* Manuel Gonzalez
* Robin Schiro

1. **Project Overview**
   1. This application uses facial detection and recognition to respond to users as they enter or exit a room. A webcam is set up to point toward the room’s entrance. When a user enters, the program can respond with an action that has been configured that user. The GUI of the application will allows users to set up profiles that specify how the program should behave. Based on this configuration, the application will communicate back and forth with the user to perform desired actions and provide useful information. If more than one person is in the room at once, the application will perform actions based on who is talking at any given time.
2. **Reference Documents**
   1. [Concept of Operations](Concept%20of%20Operations.docx)
3. **Applicable Standards**
   1. **Coding Standard**
      1. Generally, we should follow the standard defined by Microsoft for .NET programming here. However, there are a few exceptions and additions:
         1. Do not use the ‘var’ variable type. It is always best to be as unambiguous as possible when declaring variables. Using ‘var’ only increases the chance of potential confusion.
   2. **Document Standard**
      1. Normal Font: Calibri Body, 11
      2. Headings:
         1. Document Title: Calibri Body, 14, Centered at top of document
         2. Major Section: Calibri Body, 14, Bold
         3. Subsection: Calibri Body, 12, Bold
      3. Spacing: Place a line break between every major section of the document
      4. Table of Contents: Include a Table of Contents if a document is longer than 10 pages.
         1. Format: The heading of each major sections should be left aligned and the corresponding page of each major section should be right aligned. There should be a line a periods separating each heading from its corresponding page number.
      5. Modification history: A table with columns labeled Version, Date, Who, and Comment will be placed at the top of every document to display the document’s modification history. Because the table has a ‘Who’ column, we do not need to maintain a separate list of authors’ names.
   3. **Artifact Size Metric Standard**
      1. Our team will use Size Points to determine the difficulty/amount of work involved in accomplishing a task. A Size Point (SP) corresponds to a realistic amount of time that any one of us might spend on the project in one day: 2 hours. 1 Size Point can be equivalent to 2 hours of work by a person with our average amount of experience. For example, a class that would take the average person on our team 8 hours to write would be measured at 4 SP.
4. **Project Team Organization**
   1. Our group members are Robin Schiro, Manuel Gonzalez, Julian Rojas, and Jimmy Lam. We will take part in all aspects of the project, however, we will each focus on a feature of the project. Robin will be our project manager.
   2. **Responsibilities:**
      1. Robin –input from camera and algorithm for facial recognition, user interface, documentation
      2. Manuel – Speech construction AI so that it talks to user, documentation
      3. Julian – Primary action module (APIs, weather related info, give info from relevant websites to user), submodule to detect person talking, documentation
      4. Jimmy – Speech recognition, other action modules (open/close applications, logging in/out), documentation
   3. We will communicate in person as well as virtually through google hangouts if physical meetings aren’t possible. We will meet at least once a week to discuss the responsibilities of each group member and to check on everyone’s status and the status of the project.
5. **Deliverables**
6. **Software Life Cycle Process**
   1. Our group will adhere to a form of the Agile development process over the course of this project. This process divides development into a series of ‘sprints’ during which developers complete tasks that were assigned to them at the beginning of each sprint. Agile is very flexible due to ability of customers and developers to report bugs and create user stories during each sprint. We would like to gain experience using this process because of its ubiquity in the industry of software engineering and its proven success.
   2. We will have two-week sprints. At the beginning of each sprint, we will create several user stories and assign them to each team member.
   3. Weekly Status meetings- Describe what you did during the week, problems you had, what you plan to work on during the next week.
   4. Diagram:

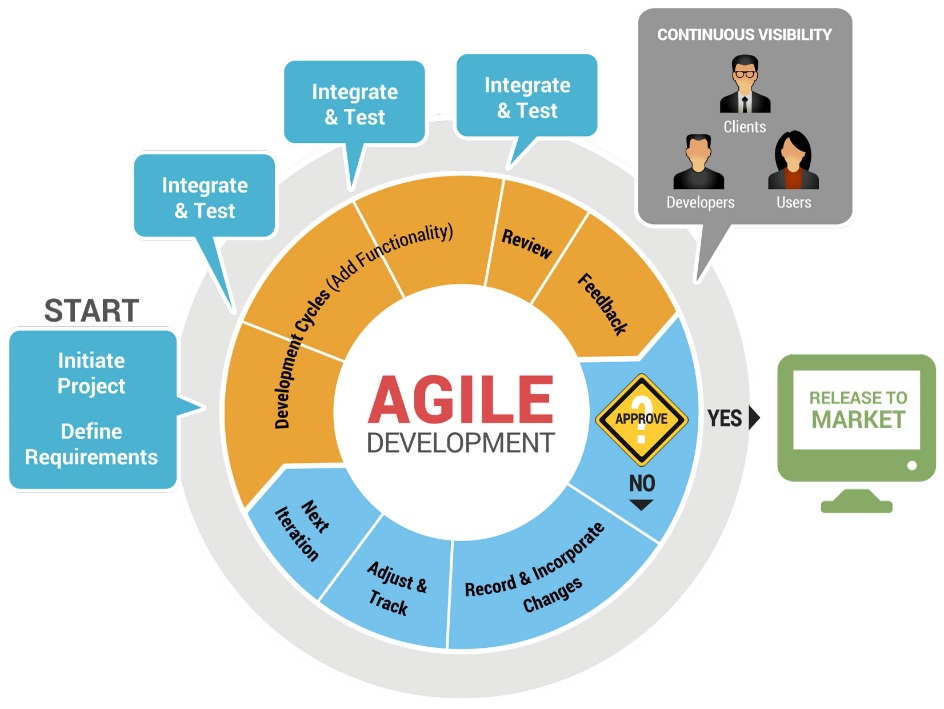
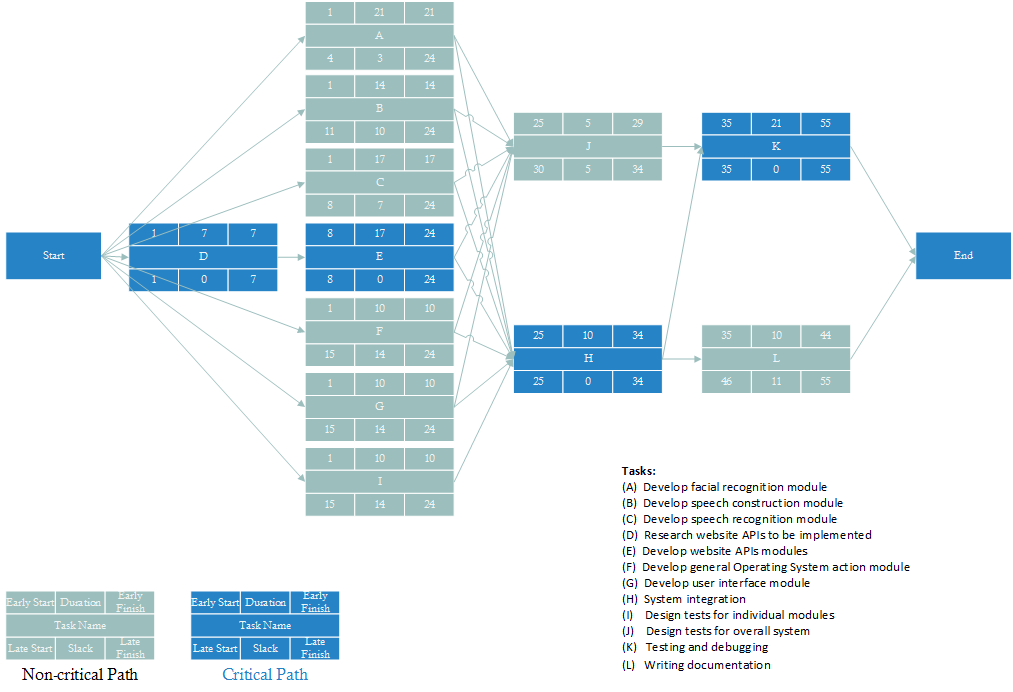


Figure 1: Representation of the Agile development cycle. Source: [STAGroup](http://www.stagrp.com/technology/application-solutions/)

1. **Tools and Computing Environment**
2. **Configuration Management**
   1. **Version Control**
      1. We will use Microsoft Visual Studio’s Team Foundation Server as a repository for our code and documentation. This repository will be used in conjunction with Git to manage checkouts and commits.
      2. Each member of the team will be responsible for minimizing conflict between pushes to the repository. Communication with the other members of the team is key in order to guarantee the stability of the repository.
   2. **Commit Process**
      1. Before working on an issue, the developer must create a branch for that issue. This is done to prevent conflict when multiple developers are working on the project simultaneously.
      2. As the developer works on an issue, he should make several commits (depending on the size of the issue) to document the history of changes he makes to the relevant code files.
      3. Once the developer has resolved the issue, he should merge his local branch with the main branch. He must be sure to communicate with the rest of the team before performing a merge.
3. **Quality Assurance**
   1. Several tests will be performed after each individual module meets the basic requirements for functioning. First starting with the common inputs that a module should expect on a normal use, and then using more extreme cases and multiple streams of data to test stress on each module.
   2. After integrating the modules, tests should be performed on data transfer between the modules. Again, a big load of common data and some extreme cases will be used to guarantee the stability of the system.
   3. Ultimately, general tests for overall performance will ensure user comfort and accessibility, as well as introducing multiple users to test accuracy when handling these type of situations.
   4. Manuel will ensure to design the tests for each module and the whole system. He will perform the tests once each team member is ready, and any failed tests will be reported in Git's issue tracking.
4. **Risk Management**
5. **Table of Work Packages, Time Estimates, and Assignments**

| Work package | Time Estimate | Responsible |
| --- | --- | --- |
| Develop facial recognition module | 3 weeks | Robin Schiro |
| Develop speech construction module | 2 weeks | Manuel Gonzalez |
| Develop speech recognition module | 2 ½ weeks | Jimmy Lam |
| Research website APIs to be implemented | 1 week | Julian Rojas |
| Develop website APIs modules | 2 ½ weeks | Julian Rojas |
| Develop general Operating System action module | 1 week | Jimmy Lam |
| Develop user interface module | 1 week | Robin Schiro |
| System integration | 1 week | Everyone |
| Design tests for individual modules and overall system | 1 week | Manuel Gonzalez |
| Testing and debugging | 3 weeks | Everyone |
| Writing documentation | 3 weeks | Everyone |

1. **PERT Chart**

1. **Technical Progress Metrics**

|  |  |
| --- | --- |
| **Metrics** | **Measure** |
| **Requirements** |  |
| Number of Requirements |  |
| Number of Requirement changes |  |
| **Specifications** |  |
| Number of Specifications |  |
| Number of Specification Changes |  |
| **Analysis and Design Phase** |  |
| UML diagrams completed |  |
| **Implementation** |  |
| Number of Packages |  |
| Number of Classes |  |
| Number of Methods |  |
| Number of Lines of Code |  |
| Number of Test Cases |  |
| **Integration** |  |
| Number of Test Cases |  |
| Number of Defects |  |

1. **Plan for tracking, control, and reporting of progress**
   1. We will individually submit a weekly report to keep track of our own progress (number of hours working on the project, responsibility report), but we will also have weekly meetings to keep track of each other’s progress and the status of the project overall. Our project manager will look at our progress reports, our technical progress metrics, reassess project risks, and our project plan to determine whether we are ahead, on time, or behind schedule. With this information, we will update our PERT chart, the technical progress metric, and update everyone with any changes to the project plan.