Software Requirements Specification

for

Input Multiplexer (iMux)

Version 1.0 approved

Prepared by Christopher Hartz

National University: SEN601

06 Aug 2017

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 2

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 3

2.4 Operating Environment 3

2.5 Design and Implementation Constraints 3

2.6 User Documentation 3

2.7 Assumptions and Dependencies 3

3. External Interface Requirements 3

3.1 User Interfaces 3

3.2 Hardware Interfaces 4

3.3 Software Interfaces 4

3.4 Communications Interfaces 4

4. System Features 4

4.1 System Feature 1 4

4.2 System Feature 2 4

5. Other Nonfunctional Requirements 5

5.1 Performance Requirements 5

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

6. Future Developments 6

Appendix A: Glossary 6

Appendix B: Analysis Models 7

Appendix C: Sample Figures 11

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Date | Reason for Changes | Version |
| Hartz | 25 July 2017 | Initial Document | 1.0 |
| Hartz | 1 Aug 2017 | Added Diagrams | 1.0 |
| Hartz | 6 Aug 2017 | Updated Functional Requirements | 1.0 |

# Introduction

## Purpose

This Document defines the software requirement for Input Multiplexer (iMux), A Google AI Cloud Services Interface Application. This document presents a detailed description of the Input Multiplexer application. It will explain the purpose and features of the system, the interfaces of the system, how it will handle input, the functions of the system, and the constraints under which it must operate. This document is intended for both the stakeholders and the developers of the system and will serve as a proof of concept for later increase in scope and follow on development.

## Document Conventions

No conventions or assumption have been used in this document. All priorities and requirements will be explicitly stated.

## Intended Audience and Reading Suggestions

This Document is intended to be a living document that serves as a communication method for all stakeholders. During each spiral of development, of which this is the first, identified stakeholders including project designers, programmers, clients, and users will use this document as a reference.

## Product Scope

iMux is the underpinning software that will enable a highly responsive and easy to use user interfaces for applications which have hard time requirements and rely heavily on human to human, and human to machine interaction in a command and control environment.

iMux will integrate a multitude of different human users (Appendix B, Fig. 4), who are using a different input methods (i.e. voice, graphical, or text), and exercising different roles (i.e. supervisor or technician), to work on a single application seamlessly. The software will lay the foundation by using Google’s AI services to integrate AI agents as additional actors in a workflow in order to facilitate complex interactions among humans, identify patterns, and automate tasks.

This Document will address the first demonstration prototype which includes a subset of the features listed above. Version 1.0 will:

1. Enable two or more people to work with one application simultaneously. One user will give voice commands and one or more users will issue text and GUI based commands. (Appendix B, Fig 3)
2. The project will implement the software in Python, using Google’s Python Style Guide, Google’s Speech Recognition Service, and Google's Voice to Text AI service.
3. A game will be modified to implement the iMux application to allow for multiple inputs and to simulate an environment with hard time requirements. (Appendix B, Fig 2)

## Coding and Style References

1. Google Python Style Guide: <https://google.github.io/styleguide/pyguide.html>
2. Pygame; Python GUI: <https://www.pygame.org/docs/>
3. Tensor Flow: <https://www.tensorflow.org/>
4. Google Cloud Tutorials: <https://www.youtube.com/playlist?list=PLQVvvaa0QuDfGVb3yucqvKtUgwOJgZWCm>
5. Python Speech Recognition: <https://pypi.python.org/pypi/SpeechRecognition/>
6. Game code contributions: [http://pygame.org/project-python+pygame+tower+defence-1296-4410.html#](http://pygame.org/project-python+pygame+tower+defence-1296-4410.html)

# Overall Description

## Product Perspective

iMux is an application that address a recent paradigm shift in computing. Microsoft, Amazon and Google have begun to invest heavily in the growth of cloud infrastructure and AI services. With these services, they believe they can revolutionize a broad range of businesses across the entire economy. Agriculture and Technology or “Agtech” is an example where these cloud services are revolutionizing a business not traditionally thought of as high tech. Thomas Grose (Grose2015) describes the trend:

“Thanks to the growing availability and falling costs of a panoply of technologies, including mobile, wireless telecoms; sensors; cloud storage of massive data sets; superfast, powerful computers capable of running complex algorithms that chomp through data to find useful, actionable information; drones; satellite guidance and mapping; robotics; and 3-D cameras.”

AI and cloud services not only bring down the cost but provide functions that cannot be replicated in the field. The computational tasks required are outside the resources available locally, which leads to a cloud centric approach. In a study of hierarchical systems of robots Gomes notes that coordination requires that their cognitive functions be moved to the cloud:

“[Systems] can benefit from cloud computing by transferring algorithms that demand high computational power to the cloud, by having access to a much larger (shared) knowledge base that can be stored in the cloud, or to have a place where they can retrieve and store new information.” (Gomes2014)

A commercial example of this appeared in Forbes. A recycling center uses a camera and a robotic arm to sort trash, speeding up the process and reducing costs ([link](https://www.forbes.com/sites/jenniferhicks/2017/04/04/this-recycling-robot-uses-artificial-intelligence-to-sort-your-recyclables/#4a8221a42d35))

“Their system used a combination of computer vision, machine learning, artificial intelligence (AI) to run synchronized robotic arms to sort and pick recycled materials from moving conveyor belts... The Clarke system has the ability to sort at super-human speeds”

These trends in computing are in essence creating inexpensive Command and Control (C2) services. In the cases listed above it is for machines but the use case also extends to environments that must integrate large teams of humans along with machines and have hard time requirements such as air traffic control, disaster response, military operations, and cyber intrusion mitigation.

iMux aims to integrate the human element into this cloud C2 paradigm in order to increase the efficiency and effectiveness of operating in these mixed AI and human hard time environments.

## Product Functions

iMux will be build using a MacBook running python 2.7 and Google’s Cloud API, using three front end interfaces on a single terminal. One interface per input type: Text, Voice, and GUI. Voice corresponding to the Supervisor and Text/GUI corresponding to the Technician. This is illustrated in Appendix B Fig 1

Data for version 1.0 will be stored on the MacBook. Subsequent versions will include data acquisition and storage implemented in the cloud.

## User Classes and Characteristics

In a paper C J Weinstein **(Weinstein1995**) identified seven user classifications in a naval air operations and correlated them their job function that could benefit from voice and text integration technologies. They were:

Soldier – Data Entry, Data Access, C2, Training, Translation

Naval CIC Officer – Data Entry, Data Access, C2, Training

Pilot – Data Entry, Data Access, Command and Control

Agent – Data Entry, Data Access, and Translation

Air Traffic Controller – Data Entry, Data Access, Training

Diplomat – Data Entry, Data Access, Translation

Joint Force Commander – Data Access, C2, Translation

In version 1.0 iMux will implement two of the above roles. The two user classes will correspond to Soldier and Joint Force Commander and will be:

1. Supervisor
2. Technician

Supervisors will interact via Voice and Technicians will use Text and GUI inputs.

## Operating Environment

iMux v 1.0 will operate in a benign environment designed to demo basic capability. It will use any Computing Platform (Mac, Windows, or Linux) running Python 2.7 with the Google Cloud API and Pygame libraries installed. User’s will interact with the computing client on a single computer.

## Design and Implementation Constraints

iMux is designed to facilitate communication between a variety of roles in a control room like setting. It must coexist in an environment with a number of other communication technologies such as chat and VoIP as well as face to face communication and still provide added value. To do this it must be error free and responsive. iMux must have a low latency and between commands and execution while implementing multiple threads of execution.

## User Documentation

Written reference cards and video documentation will be created for each role. Code will be documented during development

## Assumptions and Dependencies

The following assumptions will be made while developing iMux

1. User will have a low latency connection to Google’s Cloud Services
2. User will have the requisite python libraries installed
3. User’s environment will be low enough noise in order to enable speech recognition
4. The client computer will not have other applications that interfere with iMux’s use of the microphone

# External Interface Requirements

## User Interfaces

There are three external interfaces: voice, text, and graphical. The voice interface will initialize and wait for a key press to receive and process a voice command. Text feedback will indicate a successfully processed command or an error. The text interface will be from a console window. It will prompt the user for typed commands and echo back the performed action. The graphical interface will accept mouse’s command and provide graphical feedback to input and provide all actors a depiction of the state of the game. Examples of the interface interaction can be found in Appendix C

## Hardware Interfaces

Version 1.0 will be run from a single MacBook with the Python framework installed. The application will run on top of the Python interpreter.

## Software Interfaces

Version 1.0 will not have external software interfaces other than to implement the speech to text conversion as part of the Google Speech to Text API.

## Communications Interfaces

iMux will use the Google Speech to Text API which makes https requests. Security is not a constraint for version 1.0.

# System v 1.0 Features

## Use Case 1: Supervisor Speech to Text command

4.1.1 Description and Priority

The Supervisor’s Speech to Text function seeks to enable seamless integration of high level commands into the workflow of commands issued by other users of the software. This prototype will demonstrate the first step in creating this functionality. Upon pressing a key iMux will record and transcribe the speech to text and insert the resulting command into a queue along with other command input. This feature is the number one priority.

4.1.2 Stimulus/Response Sequences

The Supervisor must depress a key to issue the command and then speak the desired command. The supervisor will have two seconds to issue the command. (Appendix C Fig. 2)

## User Case 2: Technician GUI Input

4.2.1 Description and Priority

The Technician’s GUI input will be issued via a mouse and allow all functions of the game to be performed. This functionality is priority number two.

4.2.2 Stimulus/Response Sequences

The Technician can issue all available commands via the graphical interface (Appendix C Fig 1).

## Use Case 3: Technician Text Input

4.3.1 Description and Priority

The Technician’s Text input will appear as the standard console window (Appendix C, Fig 3). Commands will be issued in this window by typing them at a prompt. Acknowledgements and command from all three input types will be echoed to this window as to allow the technician to track interactions closely in case they warrant further interaction. This interaction will be last in the priority list

4.3.2 Stimulus/Response Sequences

The Technician will enter commands into a standard console interface window. Command List will include all available game functions

# Other Nonfunctional Requirements

## Performance Requirements

a speech-based human-machine interface can provide a natural human-like interface provided that the technology sufficiently localizes the speaker in a noisy environment (Lim2015) and has a low enough latency. iMux will use a QoS Requirements for Voice, of ≤ 150 ms of one-way latency from between command issue and command implementation (ITU G.114 standard) and localizing technologies provided by the Google API

## Safety Requirements

None.

## Security Requirements

All communication is local and will rely on the local security policy of the local infrastructure. Connection to Google Cloud services will be via HTTPS and require an SSL certificate

## Software Quality Attributes

As iMux v 1.0 is the first prototype the software must demonstrate Flexibility. Flexibility will be demonstrated by using simple data classes, simple boundaries and interfaces between classes and overall seek to minimize the number of classes.

# System v 1.1 Features

## Use Case 4: Suggested Actions

4.1.1 Description and Priority

The AI suggestion functionality will first attempt to detect patterns in the input and output streams. It will use Google’s TensorFlow ML tool to classify and weigh patterns of input commands against the current game’s state. TensorFlow provides a flexible platform for research purposes while also allowing its models to be deployed productively (Schrimpf2016). An AI actor will then suggest that course of action to the Supervisor.

4.1.2 Stimulus/Response Sequences

This function will appear as an AI actor in the command line instruction stream. Suggestion will be made on a display to the supervisor who can then decide on issuing the instructions or ignoring them.

Appendix A: Glossary

|  |  |
| --- | --- |
| Term | Definition |
| GUI | Graphical User Interface. Commands can be issued via the GUI and graphical feedback will be displayed in the GUI |
| Voice Command | Spoken command that consists of a pre-configured set of functions |
| API | Application Program interface |
| Hard Time | Refers to a scenario that is not tolerant of delay or wrongly ordered commands |
| QoS | Quality of Service |
| Technician | A technically skilled member of a team that works directly on interpreting and implementing instructions from supervisors |
| Supervisor | A user in charge of strategic decisions and instructing technicians on actions |
| User | Anyone who uses the software |

Appendix B: Analysis Models

Fig. 1: Activity Diagram

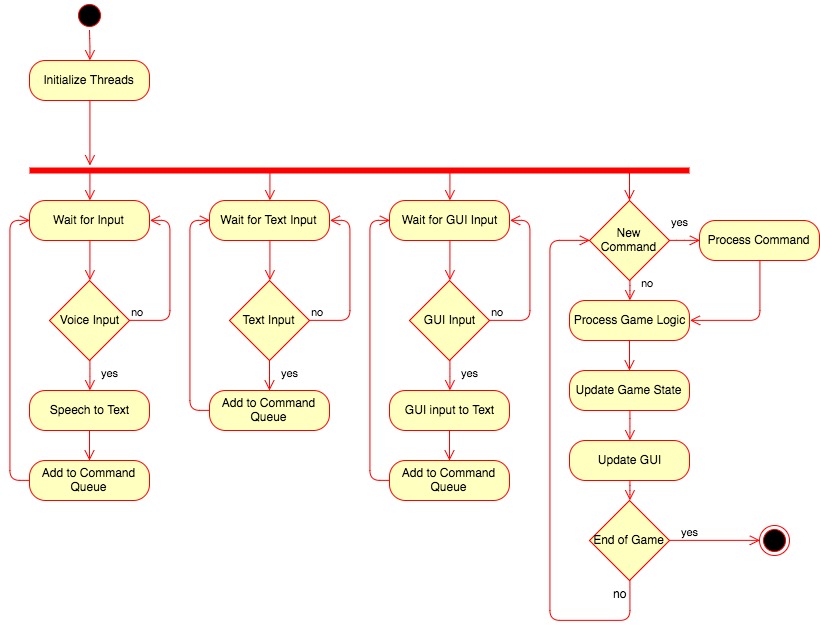


Fig 2: Class Diagram

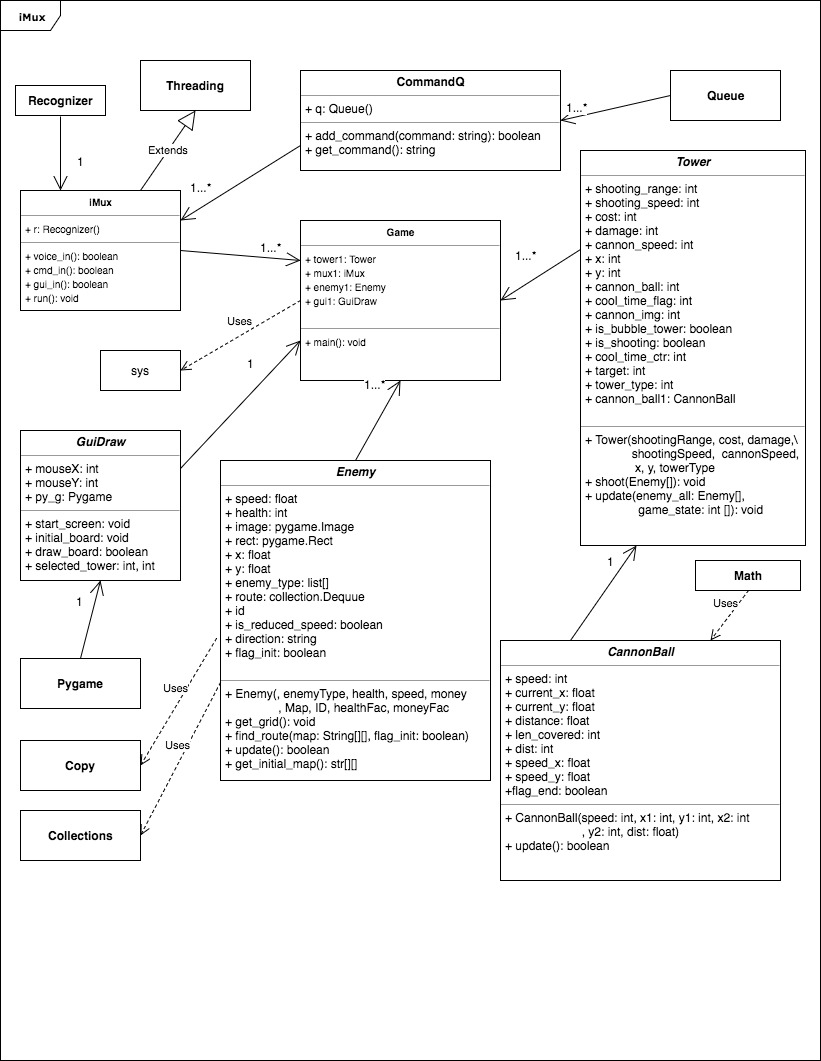


Fig 3. Sequence Diagram

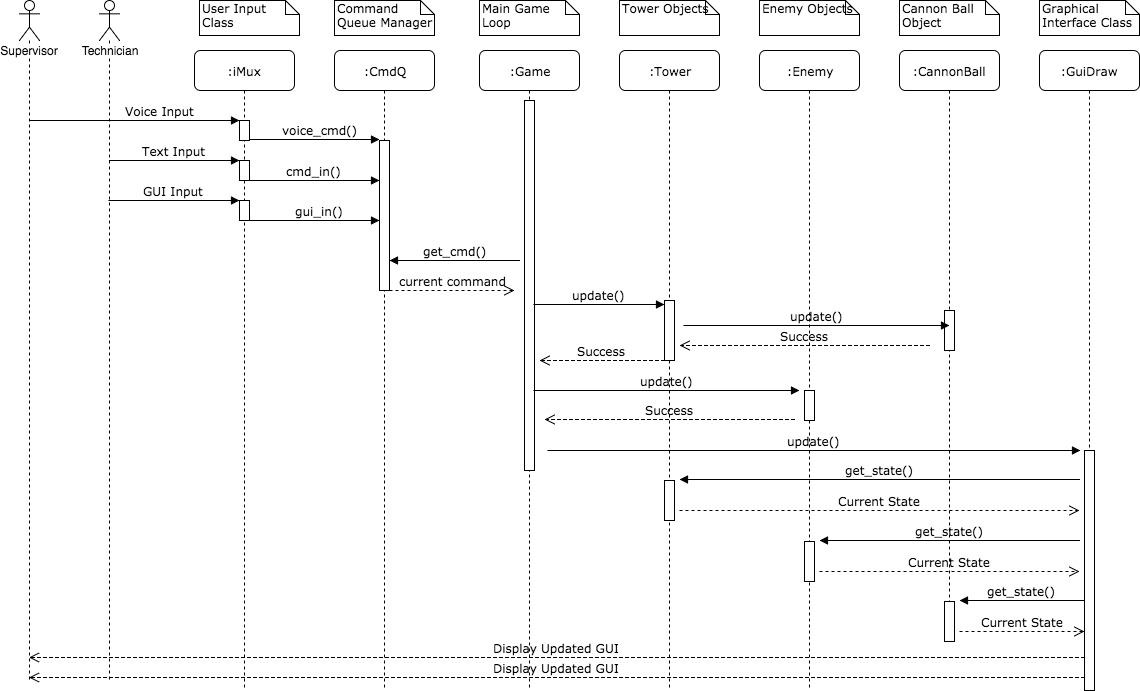
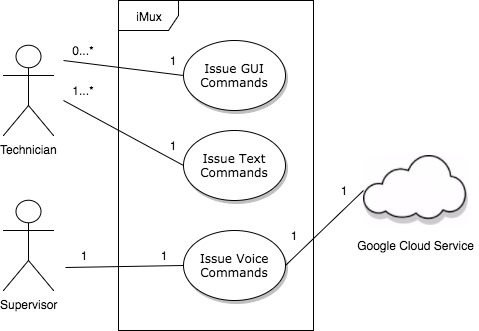


Fig 4: Use Case Diagram



Appendix C: Additional Figures

Fig. 1: Sample GUI interface



Fig. 2: Sample Voice command line feedback

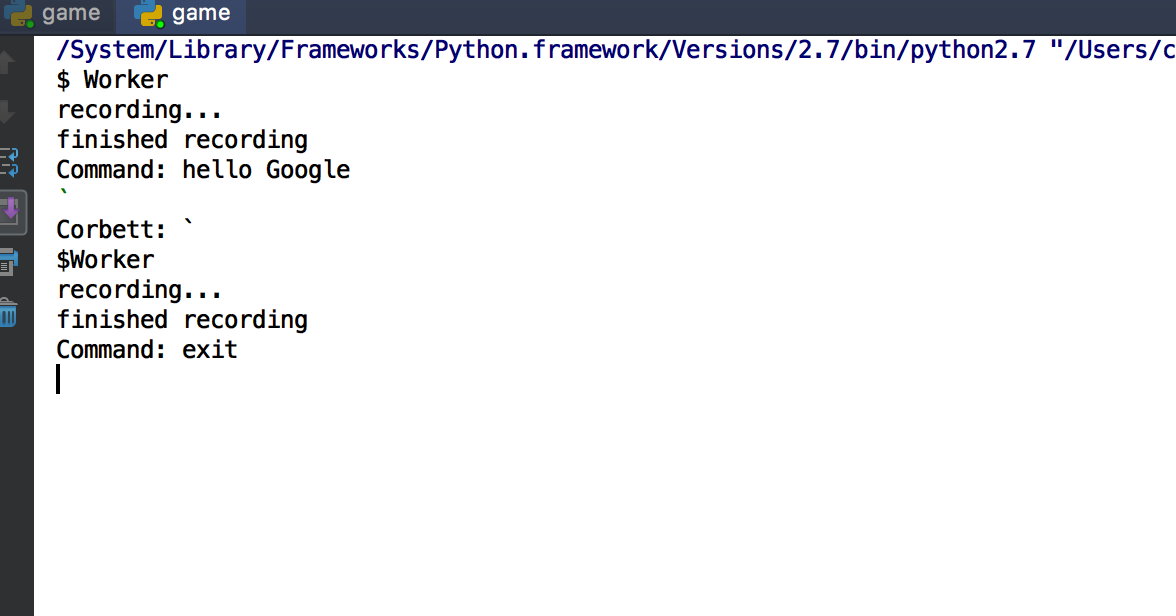


Fig. 3: Sample Text Input console command



References

**(Grose2015**) Grose, T.K (2015). The Next GREEN Revolution. ASEE Prism, 25(4), 28-31

**(Gomes2014)** Gomes, Diogo (2014). Cloud Thinking as an Intelligent Infrastructure for Mobile Robotics. Wireless Personal Communications, 76(2), 231-244

**(Weinstein1995**) C J Weinstein (1995). Military and government applications of human-machine communications by voic. PNAS 1995, 92(22) 10011-10016

**(Lim2015**) Grose, T.K (2015). H. Lim, I. C. Yoo, Y. Cho and D. Yook (2015), "Speaker localization in noisy environments using steered response voice power," in IEEE Transactions on Consumer Electronics, 61(1), 112-118

**(Schrimpf2016)**Schrimpf, Martin (2016). Should I use TensorFlow ARXIV, 7-9