Next-Step: Software Testing Plan

Jett Koele
Benjamin Huntoon
Naima Ontiveros
Kendall Callison

Sponsored by:
Jack R Williams & Zachary Lerner

Mentored by:
Scott Larocca

Table of Contents

1. Introduction

Millions of people in the U.S. struggle with mobility issues that make walking or staying active hard. Physical therapy helps improve movement and strength, but many patients lose motivation during long or repetitive exercises. This often leads to incomplete therapy and slower recovery. Our capstone project focuses on solving this problem by making physical therapy more engaging. We're building a gamified rehabilitation tool with OpenExo, an open-source exoskeleton system created by the Biomechatronics Lab at NAU. Our tool uses real-time data from sensors to let users play simple games that respond to their movement. These games are made with Unity and Pygame and help make rehab feel more like play than work.

We have created a software testing plan to check that all parts of the system perform properly. Testing helps us ensure the system does what it's supposed to, works in different situations, and is easy to use. Since this tool will be used during physical therapy, it's important that it responds quickly and gives clear feedback. Testing also helps us find and fix problems early so the system is reliable for both patients and researchers.

Our plan includes three main types of testing: integration testing, usability testing, and limited unit testing. Since we are building on top of an existing system, most of the core functionality including sensor communication and data handling has already been developed and tested. Because of this, we are not focusing heavily on unit testing. Instead, we are putting more emphasis on integration testing to make sure our added components work smoothly with the existing system. We are also focusing on usability testing to ensure the games are intuitive, fun, and helpful for therapy.

Because our project depends on real-time interaction and user engagement, integration and usability are the most important parts of our testing plan. The system must respond correctly to sensor input and be enjoyable and clear to use. The following sections explain how we will approach each type of testing, including the tools and strategies we will use.

2. Unit Testing

To start, unit testing will be conducted to ensure our backend is working as intended and properly getting data and outputting the correct data. For this we will specifically focus on our virtual gamepad part of the program and will simulate fake data from the exoskeleton. Once the fake data is processed our testing software will test for a gamepad input and ensure it outputs the correct output.

Let's take a look at what specific functions will be tested and how we will execute those tests. The first function to look at will be our basic function that translates exoskeleton data directly to controller data. To test this we will feed data into the function and ensure it translates

it to the expected value output and expected controller output. To do this we will have it return the value and use the pygame library to look for controller output from the function. If both succeed then it will move onto the next test and so forth.

The next function we will test is the scaling function. This will work similarly to the previous function but instead of outputting controller data, it will focus on the value it returns after it has been scaled. The purpose of this function is to have the researcher able to assist the user or make it harder for the user to achieve the set threshold. We will ensure that all inputs have expected outputs and the correct value is retrieved everytime.

By the end of these tests we will be able to confidently say that this part of the project is well tested and most of the expected and unexpected outcomes have been tested and documented for the future of this open source software. For each test we will ensure that the tests are conducted in a controlled environment and each output and input will be recorded and monitored as tests are being conducted. Once this step is completed we can move on to the next part of the testing.

3. Integration Testing

This section explains how we will test the different parts of our system and ensure that they work together correctly. To start, each game and the Python application will be run on different operating systems and ensure they run as intended. Once this has been achieved we can move on to some more detailed testing on each operating system.

First, let's look at how integration testing will be done for the Python application. The application will be tested by launching it and observing how fast it launches. It will also ensure that it can launch games, handle exoskeleton inputs, and scale them accordingly to controller data. The focus will be on stability, reliability, and consistency.

The application relies on two different connections between three components. Those components include the "Python Application", the "Virtual Controller" script, and the games themselves. Testing the communication between these components is what will make up our integration testing.

3.1 Python Application and Virtual Controller

To begin, the Python Application and Virtual controller scripts must communicate back and forth throughout the run time. This communication involves sending and receiving exo-skeleton data, user targets, and which sensors are active/in-use. Testing these communications is very trivial in our case. Using our test device/hardware, exoskeleton walking data can be replicated on demand. Paired with the test device we are using toggleable-debug terminal output. The output tells us what data is flowing through the system, and where the data

is in the system at any given time. One can find when and where data is missing, when and where it is incorrect, and when and where the data is correct.

These systems are simply a part of the same Python application, making it very easy to relay data. The architecture of the system utilizes Python's object-oriented features to create a "Virtual Controller" object within the main Python application. Data is simply encapsulated using getter and setter functions from within the "Virtual Controller" class. When integration testing, toggleable-debug terminal output was added to these getter and setter functions in order to reveal possible bugs and errors.

3.2 Virtual Controller and Games

The connection between the "Virtual Controller" script and the Python Application is the first of two lines of communication that needs testing; the second being the connection between the "Virtual Controller" script and the games themselves. This communication requires a very different approach in order to connect the two components. To make sure the "Virtual Controller" script is sending the right data, a third-party gamepad/controller tester is used. This third-party program visually shows controller inputs via charts and exact text data of exact values. When using both the third-party program and the Python Application, one can see if the "Virtual Controller" script creates a virtual controller/gamepad, and the data being input into the controller.

On the other hand, due to the design of our solution, one can test how well a game will interact with the rest of the application by simply using a hardware gamepad/controller. Since data created by a real physical controller and the data from the exoskeleton are identical, a physical controller can be used pre-integration to test how the game will react to exoskeleton movements.

3.3 Complete System

For purposes of testing completely, we again used our test device/hardware to exactly simulate exo-skeleton data running through the entire system. Using all the techniques described above, data can be viewed and visualized at every step of the system.

Lastly, the Python Application is being designed for use on any type of computer. Therefore, testing on as many devices as possible is necessary. Since the application is made in Python, code will be executed the same no matter what kind of computer it is running on. This is due to the design of Python itself; a user only needs to install the correct Python version. There is an exception to this though. When using third-party Python libraries, not all features may be executed the same from device to device. Though modifying the libraries is out of our reach, we must account for the issues they may present to us. Using different computers, we install the

Python Application and record any startup/runtime errors that may occur from one computer to another. This is probably the trickiest test to run due to requiring extensive hardware.

4. Usability/End-user Testing

Finally, for testing purposes, we will turn to usability and end-user testing to ensure our products accurately reflect what our thought process was while creating the project and that the users and researchers can accurately interpret the data being shown and the UI we have designed. Ensuring that both the user of the exoskeleton and the researcher are both properly able to interpret the data with little to no assistance from our team is going to be the goal here for every part of this section. Proper usage and readability of the software will ensure that both users are properly able to understand the software and data being transmitted through our program.

4.1 Usability Testing for Python Application

Researchers using the Python application should be able to navigate the system efficiently and perform essential functions with ease. The key usability tests include:

- Searching for games: Ensuring the researchers can quickly add and find games that are stored locally on the computer through the search function built within the application.
- Opening and closing games: Verifying that games launch and close correctly without errors or much delay while the application is running.
- Adjusting target goals and scaling: Checking that the system allows researchers to modify the target threshold which will adjust the scaling parameter accurately.

4.2 Usability Testing for Games

From both user and researcher perspectives, the usability of in-game feedback and settings adjustments is critical. The key tests include:

- Interpreting live data feedback: Ensuring users can easily understand and respond to real-time data feedback within the game.
- Adjusting setting during gameplay: Confirming that researchers can modify game settings dynamically without interrupting gameplay or causing system instability.

Through this usability testing, we aim to refine the system's interface, improve data clarity, and ensure a seamless experience for both users and researchers. This phase will help identify any areas where additional refinement is needed to make software as intuitive and effective as possible.

5. Conclusion

Our software plan is centered around integration and user testing as we are building off of an existing project. Unit tests, while useful, are not critical in this instance as we are not building code from scratch, only adding to the existing code base we were given. Centering our focus on those two areas of testing helps to ensure as smooth an addition to the existing code base as possible. Another key aspect of integration testing is the gamepad, in our testing it is crucial we

monitor how the gamepad responds to the wearable brace for our client as well as the controller inputs we provide.

The end-user tests in conjunction with usability tests are the other side of our testing strategy. Working with end users gives us a good idea of how a regular person would interact with the games we are making, demonstrating how complex or intuitive our designs are for someone not involved with our design process. In conclusion, we have planned for a variety of tests to be undertaken for our project to ensure the best possible quality for our client both in terms of technical tests and usability tests.