Synthogranic

Software Requirements Specification

Version 1

2/14/2024

Mohammed Asad

Team Lead

Lorenzo Alderiso

Programmer, AI Development and Visualization

Harkiran Bhullar

Programmer, AI Development and Testing

Revision History

Date	Description	Author	Comments
2/14/2024	Version 1	Mohammed Asad	First Revision

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date

Table of Contents

List of Figures

List of Tables

1. Introduction

<The introduction to the Software Requirement Specification (SRS) document should provide an overview of the complete SRS document. While writing this document please remember that this document should contain all of the information needed by a software engineer to adequately design and implement the software product described by the requirements listed in this document.>

1.1 Purpose

<What is the purpose of this SRS and the (intended) audience for which it is written?>

1.2 Scope

<This subsection should:

(1) Identify the software product(s) to be produced by name; for example, Host DBMS, Report Generator, etc.

(2) Explain what the software product(s) will, and, if necessary, will not do

(3) Describe the application of the software being specified. As a portion of this, it should:

(a) Describe all relevant benefits, objectives, and goals as precisely as possible. For example, to say that one goal is to provide effective reporting capabilities is not as good as saying parameter-driven, user-definable reports with a 2 hour turnaround and on-line entry of user parameters.

(b) Be consistent with similar statements in higher-level specifications (for example, the System Requirement Specification), if they exist. What is the scope of this software product?>

1.3 Definitions, Acronyms, and Abbreviations

<This subsection should provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS. This information may be provided by reference to one or more appendixes in the SRS or by reference to other documents.>

1.4 Overview

<This subsection should:

(1) Describe what the rest of the SRS contains

(2) Explain how the SRS is organized.>

2. General Description

<This section of the SRS should describe the general factors that affect the product and its requirements. This section does not state specific requirements; it only makes those requirements easier to understand.>

2.1 Product Perspective

<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major

components of the overall system, subsystem interconnections, and external interfaces can be helpful.>

2.2 Product Functions

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate is often effective.>

2.3 Users and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>

2.4 General Constraints

<This subsection of the SRS should provide a general description of any other items that will limit the designer/developer's options for designing/developing the system.>

2.5 Assumptions and Dependencies

<This subsection of the SRS should list each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements in the SRS. For example, an assumption might be that a specific operating system will be available on the hardware designated for the

software product. If, in fact, the operating system is not available, the SRS would then have to change accordingly.>

2.6 Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

3. Specific Requirements

<This will be the largest and most important section of the SRS. The customer requirements are embodied within Section 2 (functions), but this section will give the D-requirements that are used to guide the project's software design, implementation, and testing.

Each requirement in this section should be:

- Correct
- Traceable (both forward and backward to prior/future artifacts)
- Unambiguous
- Verifiable (i.e., testable)
- Prioritized (with respect to importance and/or stability)
- Complete
- Consistent (with other requirements)
- Uniquely identifiable (usually via numbering like 3.4.5.6)

Attention should be paid to the carefully organize the requirements presented in this section so that they may easily accessed and understood. Furthermore, this SRS is not the software design document, therefore one should avoid the tendency to over-constrain (and therefore design) the software project within this SRS.>

3.1 External Interface Requirements

3.1.1 User Interfaces

The prosthetic leg's computerized system operates without the need for a traditional user interface. Unlike conventional devices, this advanced technology eliminates the necessity for direct human interaction. Designed for seamless integration into daily life, the computer within the prosthetic leg autonomously manages its functions, ensuring optimal performance without requiring user intervention. This self-manageable system represents a pioneering approach to prosthetic design, offering users a level of convenience and reliability unmatched by traditional counterparts.

3.1.2 Hardware Interfaces

The hardware application, composed of Lego Bricks, will be meticulously assembled by the developers before deployment, ensuring precise construction and optimal functionality. No additional setup or configuration is necessary upon installation, as the hardware components are pre-configured to seamlessly integrate with the prosthetic leg system. Each Lego Brick is carefully selected and calibrated to meet the stringent performance requirements of the prosthetic leg, guaranteeing durability and reliability in various environmental conditions.

This prosthetic leg is designed to attach at the hip, encompassing all necessary components from hip to foot. Its modular design ensures that all components are of standard height, facilitating ease of use and compatibility with existing prosthetic accessories. Additionally, the standard height, of 2 feet, provides users with consistent and predictable functionality, promoting comfort and stability during everyday activities. The leg's comprehensive design aims to enhance mobility and quality of life for individuals with limb differences, offering a reliable and adaptable solution to meet their needs.

3.1.3 Software Interfaces

The software interface for the prosthetic leg encompasses a sophisticated neural network, trained on sample data to simulate human force at various knee angles and under different levels of gravity. This neural network, developed using the Blender application, serves as the backbone of the prosthetic leg's functionality, enabling it to dynamically adjust and respond to changes in the user's movement and environment.

Before deployment, the neural network code will undergo rigorous testing on a standard computational device to ensure accuracy and reliability. This testing phase will validate the neural network's ability to

interpret sensory inputs and generate appropriate responses in real time, thereby enhancing user safety and comfort.

Once the code is thoroughly tested and optimized, it will be ported onto a Raspberry Pi platform, providing a portable and lightweight solution for integrating the neural network into the prosthetic leg. The Raspberry Pi's compact form factor and low power consumption make it an ideal choice for embedded applications, allowing users to experience seamless mobility without compromising on performance.

Through this software interface, the prosthetic leg can intelligently adapt to the user's needs and provide enhanced functionality, ultimately improving their overall quality of life and mobility.

3.1.4 Communications Interfaces

The communication interface ensures smooth interaction between different components of the prosthetic leg, without requiring user intervention. This interface enables the leg to adjust to movements and changes in the environment, providing users with seamless mobility and comfort. Users can trust that the leg's underlying communication processes work efficiently behind the scenes, allowing them to focus on their daily activities without any technical hassle.

3.2 Functional Requirements

Before hardware integration, the machine learning model will be trained using a virtual model designed in Blender to simulate real-world training. After training, the software will be integrated into a Raspberry Pi for testing in the physical environment.

1.1. User class 1 - The Robot Tester

1.1.1. Walking in a straight line

1.1.1.1. Description and Priority

The ability to walk in a straight line is a fundamental function of the prosthetic leg system, with a priority of High importance. This feature enables users to navigate smoothly and confidently in various environments, enhancing their overall mobility and independence.

1.1.1.2. Stimulus/Response Sequences

Stimulus: The user initiates the walking motion.
Response: The prosthetic leg system detects the user's intention to walk and initiates the appropriate movement sequence.
Stimulus: User adjusts walking speed.
Response: The prosthetic leg system adapts the walking speed accordingly, ensuring a comfortable and natural gait.

1.1.1.3. Functional Requirements

REQ-WALK-01: The prosthetic leg system shall accurately detect the user's intention to walk and initiate the walking motion within [specified time frame].

REQ-WALK-02: The system shall maintain stability and balance throughout the walking motion, preventing any deviation from the intended straight-line path.

REQ-WALK-03: The system shall adjust step length and frequency dynamically based on the user's walking speed preferences, ensuring smooth and natural movement.

REQ-WALK-04: In the event of external disturbances (e.g., uneven terrain), the system shall automatically compensate to maintain a straight-line walking trajectory.

1.1.2. Walking up a hill

1.1.2.1. Description and Priority

Walking up a hill allows users to ascend inclines with ease, enhancing their mobility in various outdoor environments. This feature is of High priority as it significantly impacts the user’s ability to navigate uphill terrain safely and efficiently.

1.1.2.2. Stimulus/Response Sequences

Stimulus: User approaches an incline.
Response: The prosthetic leg system detects the change in terrain and adjusts the walking motion to accommodate the uphill slope.
Stimulus: The user increases exertion to climb the hill.
Response: The system enhances support and stability to assist the user in ascending the incline smoothly.

1.1.2.3. Functional Requirements

REQ-HILL-01: The prosthetic leg system shall accurately detect changes in terrain elevation and initiate appropriate adjustments to facilitate uphill walking.

REQ-HILL-02: The system shall provide sufficient support and traction to prevent slippage or loss of stability while ascending steep inclines.

REQ-HILL-03: In response to increased user exertion, the system shall dynamically adapt the walking motion to assist in overcoming uphill gradients effectively.

1.1.3. Walking down steps

1.1.3.1. Description and Priority

Walking down steps enables users to safely descend staircases and other elevated surfaces, enhancing their mobility in indoor and outdoor environments. This feature is of High priority as it is essential for navigating stairs with confidence and stability.

1.1.3.2. Stimulus/Response Sequences

Stimulus: The user approaches a staircase or descending slope.
Response: The prosthetic leg system detects the change in terrain and adjusts the walking motion to facilitate controlled descent.
Stimulus: The user adjusts pace or stride length while descending steps.
Response: The system dynamically adapts to changes in user movement to maintain stability and prevent missteps.

1.1.3.3. Functional Requirements

REQ-STEPS-01: The prosthetic leg system shall accurately detect changes in terrain elevation and initiate appropriate adjustments to facilitate controlled descent down steps or slopes.

REQ-STEPS-02: The system shall provide sufficient support and shock absorption to cushion the impact of each step during the descent, minimizing strain on the user's joints.

REQ-STEPS-03: In response to user adjustments in pace or stride length, the system shall dynamically adapt the walking motion to ensure stability and prevent tripping or stumbling.

1.1.4. Walking on a treadmill

1.1.4.1. Description and Priority

Walking on a treadmill allows users to engage in cardiovascular exercise or rehabilitation activities within controlled indoor environments. This feature is of Medium priority, providing users with a convenient option for maintaining fitness and mobility.

1.1.4.2. Stimulus/Response Sequences

Stimulus: The user initiates walking motion on the treadmill.
Response: The prosthetic leg system synchronizes with the treadmill's speed and incline settings to facilitate a consistent walking experience.
Stimulus: The user adjusts treadmill settings (e.g., speed, incline).

Response: The system dynamically adapts the walking motion to align with the user's chosen settings, ensuring a comfortable and effective workout.

1.1.4.3. Functional Requirements

REQ-TREADMILL-01: The prosthetic leg system shall synchronize with treadmill settings to maintain consistent walking speed and incline.

REQ-TREADMILL-02: The system shall provide real-time feedback to the user regarding treadmill settings and walking performance, enhancing user engagement and motivation.

REQ-TREADMILL-03: In response to changes in treadmill settings initiated by the user, the system shall dynamically adjust the walking motion to ensure a smooth and natural gait.

1.1.5. Walking on a dirt trail

1.1.5.1. Description and Priority

Walking on a dirt trail enables users to explore outdoor environments with uneven terrain, providing opportunities for recreation and adventure. This feature is of Medium priority, offering users flexibility in their recreational activities while maintaining stability and comfort.

1.1.5.2. Stimulus/Response Sequences

Stimulus: User transitions from flat ground to a dirt trail.
Response: The prosthetic leg system detects the change in terrain and adjusts the walking motion to accommodate uneven surfaces.
Stimulus: The user encounters obstacles or rough terrain on the dirt trail.
Response: The system dynamically adapts to changes in terrain to maintain stability and prevent tripping or stumbling.

1.1.5.3. Functional Requirements

REQ-TRAIL-01: The prosthetic leg system shall accurately detect changes in terrain elevation and surface texture to facilitate smooth walking on dirt trails.

REQ-TRAIL-02: The system shall provide sufficient support and shock absorption to cushion the impact of uneven surfaces, reducing the risk of discomfort or injury to the user.

REQ-TRAIL-03: In response to obstacles or rough terrain encountered by the user, the system shall dynamically adjust the walking motion to ensure stability and prevent tripping or stumbling.

1.1.6. Turning while walking

1.1.6.1. Description and Priority

Turning while walking allows users to change direction smoothly and efficiently, enhancing their maneuverability in various environments. This feature is of High priority as it significantly impacts the user's ability to navigate crowded spaces or change course while walking.

1.1.6.2. Stimulus/Response Sequences

Stimulus: The user initiates a turning motion while walking.
Response: The prosthetic leg system detects the change in direction and adjusts the walking motion to facilitate a smooth and controlled turn.
Stimulus: The user adjusts the turning radius or speed.
Response: The system dynamically adapts the walking motion to align with the user's chosen turning parameters, ensuring a comfortable and responsive turning experience.

1.1.6.3. Functional Requirements

REQ-TURN-01: The prosthetic leg system shall accurately detect changes in user direction and initiate appropriate adjustments to facilitate smooth turning while walking.

REQ-TURN-02: The system shall provide sufficient support and stability during turning motions to prevent loss of balance or instability.

REQ-TURN-03: In response to changes in turning parameters initiated by the user, the system shall dynamically adjust the walking motion to ensure a controlled and natural turning experience.

3.3 Use Cases

3.3.1 Use Case #1

Use Case Name	Walking in varying terrain
Reference	Section 3.2
Trigger	The user attempts to walk, thus shifting their weight
Precondition	The test robot is fully operational and can support its weight while standing upright.
Basic Path	1. The test robot is upright and stable. 2. A slight vertical force is applied to the robot, shifting its center of gravity. 3. The test robot will calculate the necessary movement of the leg to shift its weight back into a stable upright position. 4. If the force is continuously applied, the robot will continue to walk to correct the weight shift.
Alternative Paths	1. The test robot is pre-programmed to walk a certain distance 2. Upon execution of the program, the robot will begin to walk in a straight line. 3. Based on the input measurements from its sensors, it will automatically adjust to the terrain it is walking on. 4. The test robot reaches the defined distance.

Postcondition	The selected article is downloaded to the client machine.
Exception Paths	The robot fails to maintain stability and falls over.
Other	The robot will fail if it encounters an obstacle that rewuires jumping.

3.4 Non-Functional Requirements

Non-functional requirements may exist for the following attributes. Often these requirements must be achieved at a system-wide level rather than at a unit level. State the requirements in the following sections in measurable terms (e.g., 95% of transaction shall be processed in less than a second, system downtime may not exceed 1 minute per day, > 30 day MTBF value, etc.).

3.5.1 Performance

The adaptive AI system should be capable of processing input data and generating learning outcomes within a reasonable time frame, with an average inference time of less than 100 milliseconds.

The visualization of the AI model using Blender should render smoothly and efficiently, maintaining a frame rate of at least 30 frames per second (FPS) on standard hardware configurations.

3.5.2 Reliability

The AI system should demonstrate robustness and stability during training and inference phases, with an expected failure rate of less than 0.1% during real-time testing with robotic legs.
The Blender visualization tool should be reliable, with minimal crashes or errors occurring during model rendering and animation tasks.

3.5.3 Availability

The AI system should be available for training and testing purposes at least 99.9% of the time, excluding scheduled maintenance windows and updates.

The Blender visualization environment should be accessible for design and rendering tasks throughout the development lifecycle, with minimal downtime or interruptions.

3.5.4 Security

The AI system's training data and model parameters should be securely stored and encrypted to prevent unauthorized access or tampering.

Access controls should be implemented for the Blender visualization tool to restrict editing privileges and protect intellectual property rights.

3.5.5 Maintainability

The AI system's codebase and documentation should be well-organized and extensively commented to facilitate future maintenance and updates by development teams.

Regular code reviews and version control practices should be enforced to ensure code quality and traceability of changes.

3.5.6 Portability

The adaptive AI system should be designed for cross-platform compatibility, allowing seamless deployment and integration across different operating systems and hardware configurations.

The Blender visualization files should be compatible with standard 3D modeling software and rendering engines, enabling collaboration and interoperability with external tools and environments.

3.5 Design Constraints

<Specify design constrains imposed by other standards, company policies, hardware limitation, etc. that will impact this software project. Example, the software is required to have a login screen based on company policies.>

3.6 Logical Database Requirements

<Will a database be used? If so, what logical requirements exist for data formats, storage capabilities, data retention, data integrity, etc?>

3.7 Other Requirements

4. Analysis Models

4.1 Sequence Diagrams

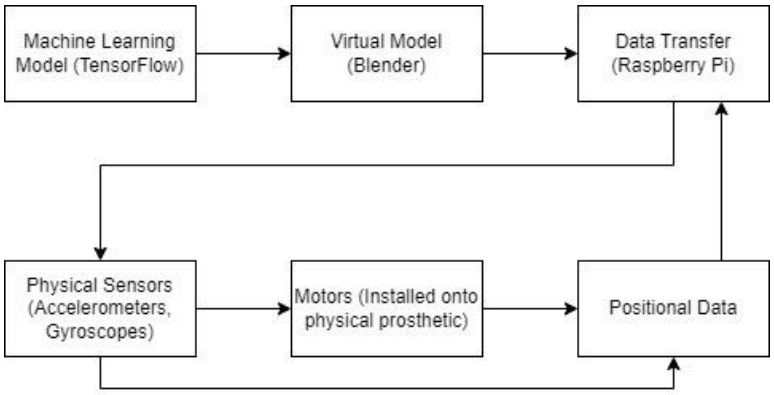

Figure 1: Reference Section 3.2

5. Change Management Process

<Identify and describe the process that will be used to update the SRS, as needed, when project scope or requirements change. Who can submit changes and by what means, and how will these changes be approved.>

References

A. Appendices

<Appendices may be used to provide additional (and hopefully helpful) information. If present, the SRS should explicitly state whether the information contained within an appendix is to be considered as a part of the SRS's overall set of requirements. Example Appendices could include

(initial) conceptual documents for the software project, marketing materials, minutes of meetings with the customer(s), etc.>

A.1 Appendix 1

A.2 Appendix 2