Synthorganic

Design Documentation

Software Design Document Specification Template

The Software Design Specification (SDS) sections provide you with guidelines related to the structure and the contents of SDS document. The Software Design Specification document includes at least these sections.

For the project, your team may have good reasons for wanting to deviate from this proposed outline. If a section is not applicable in your case, do not delete it; instead, give the topic heading and write "Not applicable".

You will note that there is some overlap in the content between different documents (i.e. the User Requirements Specification Document and the Software Design Specification Document.) This redundancy allows each document to stand on its own.

ONLY THE SECTION TITLES COLORED IN ORANGE ARE REQUIRED TO BE COMPLETED.

DO NOT DELETE THE SECTIONS YOU ARE NOT COMPLETING AS THEY ARE A PART OF THE DOCUMENT

Contents

1 Introduction

1.1 Purpose of this document

Full description of the main objectives of the SDS document.

1.2 Scope of the development project

This will be similar to what was written in the SRS.

1.3 Definitions, acronyms, and abbreviations

Be sure to alphabetize!

1.4 References

This section will include technical books and documents related to design issues. Be certain that the references you give are complete and in the appropriate format.

1.5 Overview of document

A short description of how the rest of the SDS is organized and what can be found in the rest of the document. This is not simply a table of contents. Motivate and briefly describe the various parts!

2 System architecture description

2.1 Overview of modules / components

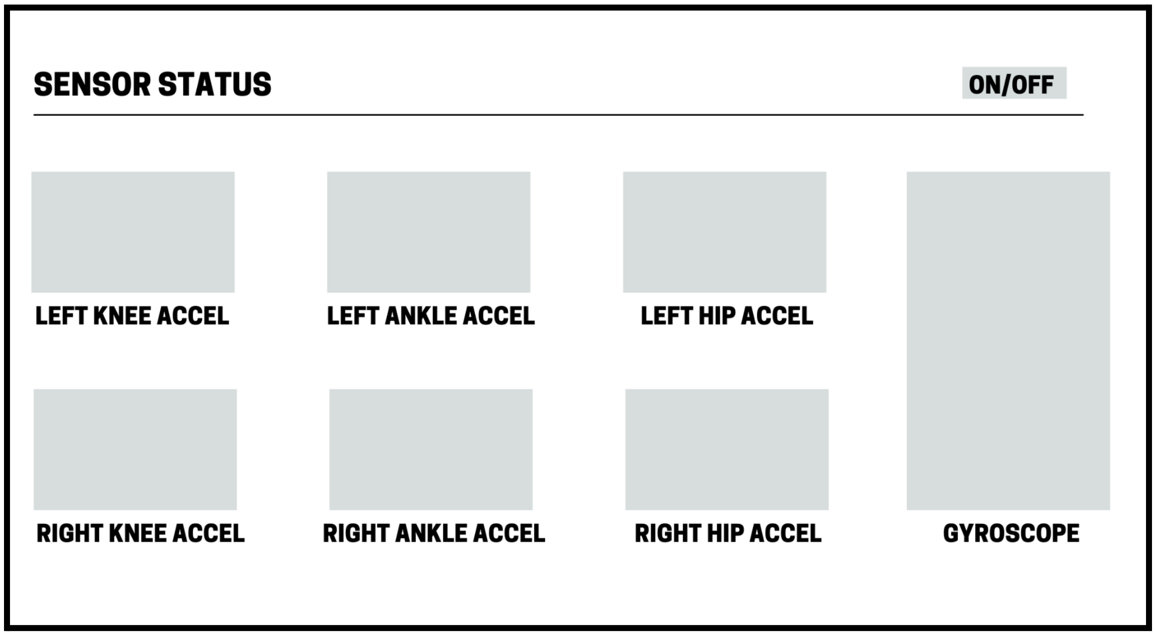
This subsection will introduce the various components and subsystems.

2.2 Structure and relationships

Make clear the interrelationships and dependencies among the various components. Structure charts can be useful here. A simple finite state machine can be useful in demonstrating the operation of the product. Include explanatory text to help the reader understand any charts.

2.3 User interface

Complete design with user interactive windows, error messages, etc. Include all windows and messages that will be viewed by the user and also the ones that the user will be interacting with.



2.4 User interface issues

The main principles of the product’s user interface include the status of the sensors on all 6 joints: knees, ankles, and hips, as well as the gyroscope and the on/off switch. The accelerometers on each of the joints calculate how fast the joint/limb is going so the AI can adjust. The status of each of these joints will show on the user interface. The gyroscope controls the balance and its status will also be displayed in the user interface. Finally, there is an on/off switch which indicates whether or not the hardware is on or off.

When a user is attempting to walk in varying terrain, the sensors would update their status to the UI. The joints would all send data back depending on the speed that they are going when walking in such terrain. The gyroscope would send data back as it is losing balance and its center of gravity is shifted because of the terrain. The joints would then adjust to the change and the sensors would send back different data as the robot corrects itself.

3 Detailed description of components (ONLY 2 ARE REQUIRED)

3.1 X Component (or Class or Function ...)

Use exactly the template shown at the end of the document.

3.2 Y Component (or Class or Function ...)

...

Examples of a component are:

1) Database
2) Server
3) Client application
4) Search Application
5) Etc.

4 Reuse and relationships to other products

For teams doing enhancement work, reuse is an important issue. Most enhancement work should focus on extending, rather than replacing, the design and product development from earlier semesters. For teams doing new development, reuse can also be an important strategy. In some cases, there is freeware that could be incorporated. In other cases, there are existing modules or classes that could be adapted. Another possibility is the use of special tools that produce open source results and thus permissible under the terms of this course.

This section should include the following subsections as appropriate:

- How reuse is playing a role in your product design
- How reuse is playing a role in your product implementation (and the motivation for changes)
- If you are not reusing material that is available, then give motivation for why it is being thrown out.

5 Design decisions and tradeoffs

Throughout the development of the Synthorganic project, several design decisions and tradeoffs are being made to ensure the effectiveness, efficiency, and safety of the AI-enhanced prosthetic leg optimization system. These decisions were motivated by a combination of technical considerations, user requirements, and practical constraints. Below are some key design decisions and tradeoffs:

Choice of TensorFlow and Blender:

- Decision: TensorFlow was selected as the primary platform for developing the AI model due to its versatility and extensive community support in the field of machine learning. Blender was chosen for visualization purposes because of its powerful 3D rendering capabilities and compatibility with TensorFlow models.
- Tradeoff: While TensorFlow and Blender offer significant advantages, there is a learning curve associated with mastering these tools. However, the benefits of using established platforms outweighed the initial learning investment.

Phased Development Approach:

- Decision: The project adopted a phased development approach, focusing on walking on flat surfaces first, followed by incline/decline surfaces, and then varying terrains. This approach allows for incremental improvements and validation at each stage.
- Tradeoff: While phased development ensures a systematic and manageable progression, it may extend the overall project timeline compared to developing all

features simultaneously. However, this tradeoff was deemed necessary to maintain quality and manage complexity effectively.

Hardware Integration Strategy:

- Decision: Integration testing was planned to involve a functional skeleton structure of robotic legs, allowing for real-world validation of the AI model's performance. This approach ensures that hardware limitations and user interactions are considered from the early stages of development.
- Tradeoff: Building a functional skeleton structure for integration testing requires additional resources and time compared to virtual simulations. However, the physical testing environment provides more accurate insights into real-world usability and performance.

Incorporation of User Feedback Loop:

- Decision: The project includes mechanisms for gathering and incorporating user feedback to improve the AI model over time. This iterative approach ensures that the system evolves to better meet user needs and preferences
- Tradeoff: Implementing a user feedback loop adds complexity to the development process and requires ongoing maintenance and support. However, the benefits of creating a user-centric solution justify the additional effort.

Balance Between Complexity and Usability:

- Decision: Throughout the design process, there was a conscious effort to balance complexity with usability, ensuring that the AI-enhanced prosthetic leg remains intuitive and accessible to users of varying technical backgrounds.
- Tradeoff: Simplifying the user interface and system interactions may require sacrificing some advanced features or customization options. However, prioritizing usability ultimately enhances user acceptance and adoption.

6 Pseudocode for components

Utilize the use cases to create pseudocode for components.

7 Appendices (if any)

<u>Software component template for section 3</u>

The template given below suggests a reasonable structure for giving a thorough description of each component described in Part 3 of the SDS. The specific information depends in part on the design approach. Your team must adapt this template to your needs and describe it in section 3.1 of your SDS.

<u>Identification</u>	The unique name for the component and the location of the component in the system.
<u>Type</u>	A module, a subprogram, a data file, a control procedure, a class, etc.
<u>Purpose</u>	Function and performance requirements implemented by the design component, including derived requirements. Derived requirements are not explicitly stated in the SRS, but are implied or adjunct to formally stated SDS requirements.
<u>Function</u>	What the component does, the transformation process, the specific inputs that are processed, the algorithms that are used, the outputs that are produced, where the data items are stored, and which data items are modified.
Subordinates	The internal structure of the component, the constituents of the component, and the functional requirements satisfied by each part.
<u>Dependencies</u>	How the component's function and performance relate to other components. How this component is used by other components. The other components that use this component. Interaction details such as timing, interaction conditions (such as order of execution and data sharing), and responsibility for creation, duplication, use, storage, and elimination of components.

<u>Interfaces</u>	Detailed descriptions of all external and internal interfaces as well as of any mechanisms for communicating through messages, parameters, or common data areas. All error messages and error codes should be identified. All screen formats, interactive messages, and other user interface components (originally defined in the SRS) should be given here.
Resources	A complete description of all resources (hardware or software) external to the component but required to carry out its functions. Some examples are CPU execution time, memory (primary, secondary, or archival), buffers, I/O channels, plotters, printers, math libraries, hardware registers, interrupt structures, and system services.
Processing	The full description of the functions presented in the Function subsection. Pseudocode can be used to document algorithms, equations, and logic.
Data	For the data internal to the component, describes the representation method, initial values, use, semantics, and format. This information will probably be recorded in the data dictionary.