# Product Title

## Senior Design Final Documentation

The Software Engineering Adventure Line

Jonathan Tomes Erik Hattervig Andrew Koc

February 17, 2014

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Mission

Mission statement inserted here.

# Document Preparation and Updates

Current Version [1.0.0]

*Prepared By:*
*Erik Hattervig*
*Andrew Koc*
*Jonathan Tomes*

***Revision History***

| ***Date*** | ***Author*** | ***Version*** | ***Comments*** |
|---|---|---|---|
| ***2/2/12*** | *Team Member #3* | *1.0.0* | *Initial version* |
| ***3/4/12*** | *Team Member #3* | *1.1.0* | *Edited version* |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# 1

# Overview and concept of operations

The overview should take the form of an executive summary. Give the reader a feel for the purpose of the document, what is contained in the document, and an idea of the purpose for the system or product.

## 1.1 Scope

What scope does this document cover?

## 1.2 Purpose

What is the purpose of the system or product?

### 1.2.1 Major System Component #1

Describe briefly the role this major component plays in this system.

### 1.2.2 Major System Component #2

Describe briefly the role this major component plays in this system.

### 1.2.3 Major System Component #3

Describe briefly the role this major component plays in this system.

## 1.3 Systems Goals

Briefly describe the overall goals this system plans to achieve. These goals are typically provided by the stakeholders. This is not intended to be a detailed requirements listing. Keep in mind that this section is still part of the Overview.

## 1.4 System Overview and Diagram

Provide a more detailed description of the major system components without getting too detailed. This section should contain a high-level block and/or flow diagram of the system highlighting the major components. See Figure 1.1. This is a floating figure environment. LaTeX will try to put it close to where it was typeset but will not allow the figure to be split if moving it can not happen. Figures, tables, algorithms and many other floating environments are automatically numbered and placed in the appropriate type of table of contents. You can move these and the numbers will update correctly.

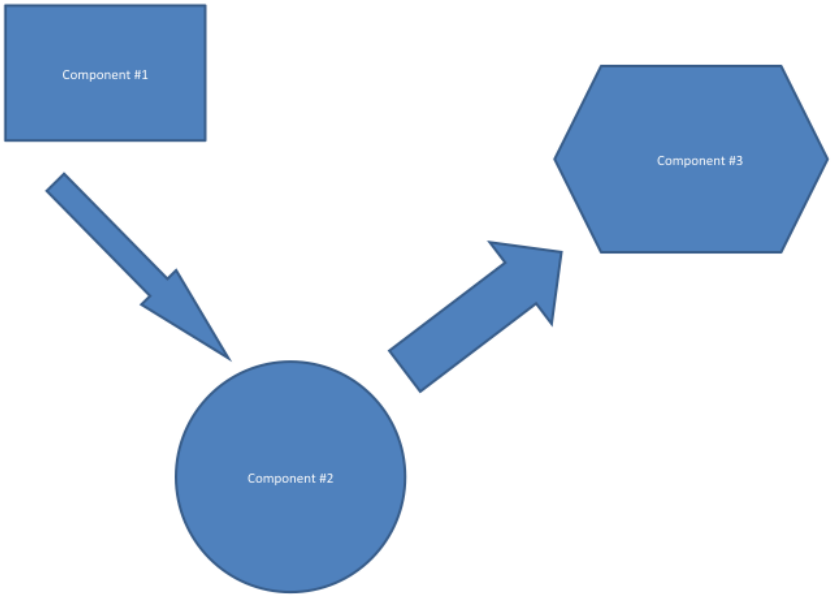


Figure 1.1: A sample figure .... System Diagram

## 1.5 Technologies Overview

This section should contain a list of specific technologies used to develop the system. The list should contain the name of the technology, brief description, link to reference material for further understanding, and briefly how/where/why it was used in the system. See Table 1.1. This is a floating table environment. LaTeX will try to put it close to where it was typeset but will not allow the table to be split.

| | |
|---:|---|
| 7C0 | hexadecimal |
| 3700 | octal |
| 11111000000 | binary |
| 1984 | decimal |

Table 1.1: A sample Table ... some numbers.

# 2

# Project Overview

This section provides some housekeeping type of information with regard to the team, project, etc.

## 2.1 Team Members and Roles

The team consists of Erik Hattervig, Andrew Koc, and Jonathon Tomes. Erik Hattervig is the Product Owner, Andrew Koc is the Technical Lead, and Jonathon Tomes is the Scrum Master. Erik is responsible for understanding the overall expectations of the product and communication with the customer about specific details regarding the operation and design of the product. Andrew is responsible for designing the technical aspects of the code. Jonathon is responsible for managing meetings and communication between team members and making sure the project is on schedule.

## 2.2 Project Management Approach

The sprint length for this project was 2 weeks. We began with a meeting to decide the user needs and split the program accordingly. Each of us would code different parts of the program and then we would all test and re-code as needed.

The code was stored, backed up, and shared through git hub. The back log and ownership was tracked through Trello. The user stories were condensed and placed on Trello to help design break points to split up the program between team members.

## 2.3 Phase Overview

The first phase of this Testing program was just to begin working on the program. The main purpose was to get to receive a root directory, find a .cpp file in the root. It would then write a log file that starting with a time stamp to be used later to record the results of tests.

After that it would a crawl through the sub directories recursively starting at the root, looking for .tst files that would be test cases for the program. Along with these would be .ans files that would allow us to compare the program output and see wich test cases failed.

It would then out put the results of each test to a log file. With a final log write that writes the percentage of passed and failed tests.

## 2.4 Terminology and Acronyms

none.

# 3

# User Stories, Backlog and Requirements

## 3.1 Overview

The overview should take the form of an executive summary. Give the reader a feel for the purpose of the document, what is contained in the document, and an idea of the purpose for the system or product.

The userstories are provided by the stakeholders. You will create he backlogs and the requirements, and document here. This chapter should contain details about each of the requirements and how the requirements are or will be satisfied in the design and implementation of the system.

Below: list, describe, and define the requirements in this chapter. There could be any number of subsections to help provide the necessary level of detail.

### 3.1.1 Scope

What scope does this document cover? This document would contain stakeholder information, initial user stories, requirements, proof of concept results, and various research task results.

### 3.1.2 Purpose of the System

What is the purpose of the system or product?

## 3.2 Stakeholder Information

This section would provide the basic description of all of the stakeholders for the project. Who has an interest in the successful and/or unsuccessful completion of this project?

### 3.2.1 Customer or End User (Product Owner)

Who? What role will they play in the project? Will this person or group manage and prioritize the product backlog? Who will they interact with on the team to drive product backlog priorities if not done directly?

### 3.2.2 Management or Instructor (Scrum Master)

Who? What role will they play in the project? Will the Scrum Master drive the Sprint Meetings?

### 3.2.3 Investors

Are there any? Who? What role will they play?

### 3.2.4 Developers –Testers

Who? Is there a defined project manager, developer, tester, designer, architect, etc.?

## 3.3 Business Need

Use this section to define what business need exist and how this software will meet and/or exceed that business need.

## 3.4 Requirements and Design Constraints

Use this section to discuss what requirements exist that deal with meeting the business need. These requirements might equate to design constraints which can take the form of system, network, and/or user constraints. Examples: Windows Server only, iOS only, slow network constraints, or no offline, local storage capabilities.

### 3.4.1 System Requirements

What are they? How will they impact the potential design? Are there alternatives?

### 3.4.2 Network Requirements

What are they?

### 3.4.3 Development Environment Requirements

What are they? Is the system supposed to be cross-platform?

### 3.4.4 Project Management Methodology

The stakeholders might restrict how the project implementation will be managed. There may be constraints on when design meetings will take place. There might be restrictions on how often progress reports need to be provided and to whom.

- What system will be used to keep track of the backlogs and sprint status?
- Will all parties have access to the Sprint and Product Backlogs?
- How many Sprints will encompass this particular project?
- How long are the Sprint Cycles?
- Are there restrictions on source control?

## 3.5 User Stories

This section can really be seen as the guts of the document. This section should be the result of discussions with the stakeholders with regard to the actual functional requirements of the software. It is the user stories that will be used in the work breakdown structure to build tasks to fill the product backlog for implementation through the sprints.

This section should contain sub-sections to define and potentially provide a breakdown of larger user stories into smaller user stories.

### 3.5.1 User Story #1

User story #1 discussed.

#### 3.5.1.a User Story #1 Breakdown

Does the first user story need some division into smaller, consumable parts by the reader? This does not need to go to the level of actual task definition and may not be required.

### 3.5.2 User Story #2

#### 3.5.2.a User Story #2 Breakdown

User story #2 ....

### 3.5.3 User Story #3

#### 3.5.3.a User Story #3 Breakdown

User story #3 ....

## 3.6 Research or Proof of Concept Results

This section is reserved for the discussion centered on any research that needed to take place before full system design. The research efforts may have led to the need to actually provide a proof of concept for approval by the stakeholders. The proof of concept might even go to the extent of a user interface design or mockups.

## 3.7 Supporting Material

This document might contain references or supporting material which should be documented and discussed either here if approprite or more often in the appendices at the end. This material may have been provided by the stakeholders or it may be material garnered from research tasks.

# 4

# Design and Implementation

This section is used to describe the design details for each of the major components in the system. This section is not brief and requires the necessary detail that can be used by the reader to truly understand the architecture and implementation details without having to dig into the code. Sample algorithm: Algorithm 1. This algorithm environment is automatically placed - meanig it floats. You don't have to worry about placement or numbering.

**Algorithm 1** Calculate $y = x^n$

**Require:** $n \geq 0 \vee x \neq 0$
**Ensure:** $y = x^n$
  $y \Leftarrow 1$
  **if** $n < 0$ **then**
    $X \Leftarrow 1/x$
    $N \Leftarrow -n$
  **else**
    $X \Leftarrow x$
    $N \Leftarrow n$
  **end if**
  **while** $N \neq 0$ **do**
    **if** $N$ is even **then**
      $X \Leftarrow X \times X$
      $N \Leftarrow N/2$
    **else** {$N$ is odd}
      $y \Leftarrow y \times X$
      $N \Leftarrow N - 1$
    **end if**
  **end while**

Citations look like [2, 1, 3] and [6, 4, 5]. These are done automatically. Just fill in the database `designrefs.bib` using the same field structure as the other entries. Then pdflatex the document, bibtex the document and pdflatex twice again (don't ask why so many times). The bibiliography is automatically constructed.

## 4.1 Major Component #1

### 4.1.1 Technologies Used

This section provides a list of technologies used for this component. The details for the technologies have already been provided in the Overview section.

### 4.1.2 Component Overview

This section can take the form of a list of features.

### 4.1.3 Phase Overview

This is an extension of the Phase Overview above, but specific to this component. It is meant to be basically a brief list with space for marking the phase status.

### 4.1.4 Architecture Diagram

It is important to build and maintain an architecture diagram. However, it may be that a component is best described visually with a data flow diagram.

### 4.1.5 Data Flow Diagram

It is important to build and maintain a data flow diagram. However, it may be that a component is best described visually with an architecture diagram.

### 4.1.6 Design Details

This is where the details are presented and may contain subsections. Here is an example code listing:

```
#include <stdio.h>
#define N 10
/* Block
 * comment */

int main()
{
    int i;

    // Line comment.
    puts("Hello world!");

    for (i = 0; i < N; i++)
    {
        puts("LaTeX is also great for programmers!");
    }

    return 0;
}
```

This code listing is not floating or automatically numbered. If you want autonumbering, but it in the algorithm environment (not algorithmic however) shown above.

## 4.2 Major Component #2

### 4.2.1 Technologies Used

This section provides a list of technologies used for this component. The details for the technologies have already been provided in the Overview section.

### 4.2.2 Component Overview

This section can take the form of a list of features.

### 4.2.3 Phase Overview

This is an extension of the Phase Overview above, but specific to this component. It is meant to be basically a brief list with space for marking the phase status.

### 4.2.4 Architecture Diagram

It is important to build and maintain an architecture diagram. However, it may be that a component is best described visually with a data flow diagram.

### 4.2.5 Data Flow Diagram

It is important to build and maintain a data flow diagram. However, it may be that a component is best described visually with an architecture diagram.

### 4.2.6 Design Details

This is where the details are presented and may contain subsections.

## 4.3 Major Component #3

### 4.3.1 Technologies Used

This section provides a list of technologies used for this component. The details for the technologies have already been provided in the Overview section.

### 4.3.2 Component Overview

This section can take the form of a list of features.

### 4.3.3 Phase Overview

This is an extension of the Phase Overview above, but specific to this component. It is meant to be basically a brief list with space for marking the phase status.

### 4.3.4 Architecture Diagram

It is important to build and maintain an architecture diagram. However, it may be that a component is best described visually with a data flow diagram.

### 4.3.5 Data Flow Diagram

It is important to build and maintain a data flow diagram. However, it may be that a component is best described visually with an architecture diagram.

### 4.3.6 Design Details

This is where the details are presented and may contain subsections.

# 5

# System and Unit Testing

This section describes the approach taken with regard to system and unit testing.

## 5.1 Overview

Provides a brief overview of the testing approach, testing frameworks, and general how testing is/will be done to provide a measure of success for the system.

## 5.2 Dependencies

Describe the basic dependencies which should include unit testing frameworks and reference material.

## 5.3 Test Setup and Execution

Describe how test cases were developed, setup, and executed. This section can be extremely involved if a complete list of test cases was warranted for the system.

# 6

# Development Environment

The basic purpose for this section is to give a developer all of the necessary information to setup their development environment to run, test, and/or develop.

## 6.1 Development IDE and Tools

Describe which IDE and provide links to installs and/or reference material.

## 6.2 Source Control

Which source control system is/was used? How was it setup? How does a developer connect to it?

## 6.3 Dependencies

Describe all dependencies associated with developing the system.

## 6.4 Build Environment

How are the packages built? Are there build scripts?

## 6.5 Development Machine Setup

If warranted, provide a list of steps and details associated with setting up a machine for use by a developer.

# 7

# Release – Setup – Deployment

This section should contain any specific subsection regarding specifics in releasing, setup, and/or deployment of the system.

## 7.1 Deployment Information and Dependencies

Are there dependencies that are not embedded into the system install?

## 7.2 Setup Information

How is a setup/install built?

## 7.3 System Versioning Information

How is the system versioned?

# 8

# User Documentation

This section should contain the basis for any end user documentation for the system. End user documentation would cover the basic steps for setup and use of the system. It is likely that the majority of this section would be present in its own document to be delivered to the end user. However, it is recommended the original is contained and maintained in this document.

## 8.1 User Guide

The source for the user guide can go here. You have some options for how to handle the user docs. If you have some `newpage` commands around the guide then you can just print out those pages. If a different formatting is required, then have the source in a separate file `userguide.tex` and include that file here. That file can also be included into a driver (like the senior design template) which has the client specified formatting. Again, this is a single source approach.

## 8.2 Installation Guide

## 8.3 Programmer Manual

# 9

# Class Index

## 9.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

# 10

# Class Documentation

## 10.1 Poly Class Reference

### Public Member Functions

- Poly ()
- ~Poly ()
- int myfunction (int)

### 10.1.1 Constructor & Destructor Documentation

#### 10.1.1.a Poly::Poly ( )

My constructor

#### 10.1.1.b Poly::~Poly ( )

My destructor

### 10.1.2 Member Function Documentation

#### 10.1.2.a int Poly::myfunction ( int *a* )

my own example function fancy new function
new variable
The documentation for this class was generated from the following file:

- hello.cpp

# Acknowledgement

Thanks

# Supporting Materials

This document will contain several appendices used as a way to separate out major component details, logic details, or tables of information. Use of this structure will help keep the document clean, readable, and organized.

# Sprint Reports

## 10.1 Sprint Report #1

## 10.2 Sprint Report #2

## 10.3 Sprint Report #3

# Industrial Experience

## 10.4 Resumes

## 10.5 Industrial Experience Reports

### 10.5.1 Name1

### 10.5.2 Name2

### 10.5.3 Name3

# Appendix

Latex sample file:

## 10.1 Introduction

This is a sample input file. Comparing it with the output it generates can show you how to produce a simple document of your own.

## 10.2 Ordinary Text

The ends of words and sentences are marked by spaces. It doesn't matter how many spaces you type; one is as good as 100. The end of a line counts as a space.

One or more blank lines denote the end of a paragraph.

Since any number of consecutive spaces are treated like a single one, the formatting of the input file makes no difference to TEX, but it makes a difference to you. When you use LATEX, making your input file as easy to read as possible will be a great help as you write your document and when you change it. This sample file shows how you can add comments to your own input file.

Because printing is different from typewriting, there are a number of things that you have to do differently when preparing an input file than if you were just typing the document directly. Quotation marks like "this" have to be handled specially, as do quotes within quotes: "'this' is what I just wrote, not 'that'".

Dashes come in three sizes: an intra-word dash, a medium dash for number ranges like 1–2, and a punctuation dash—like this.

A sentence-ending space should be larger than the space between words within a sentence. You sometimes have to type special commands in conjunction with punctuation characters to get this right, as in the following sentence. Gnats, gnus, etc. all begin with G. You should check the spaces after periods when reading your output to make sure you haven't forgotten any special cases. Generating an ellipsis ... with the right spacing around the periods requires a special command.

TEX interprets some common characters as commands, so you must type special commands to generate them. These characters include the following: $ & % # { and }.

In printing, text is emphasized by using an *italic* type style.

*A long segment of text can also be emphasized in this way. Text within such a segment given additional emphasis with* Roman *type. Italic type loses its ability to emphasize and become simply distracting when used excessively.*

It is sometimes necessary to prevent TEX from breaking a line where it might otherwise do so. This may be at a space, as between the "Mr." and "Jones" in "Mr. Jones", or within a word—especially when the word is a symbol like *itemnum* that makes little sense when hyphenated across lines.

Footnotes[1] pose no problem.

TEX is good at typesetting mathematical formulas like $x - 3y = 7$ or $a_1 > x^{2n}/y^{2n} > x'$. Remember that a letter like $x$ is a formula when it denotes a mathematical symbol, and should be treated as one.

[1] This is an example of a footnote.

## 10.3 Displayed Text

Text is displayed by indenting it from the left margin. Quotations are commonly displayed. There are short quotations

> This is a short a quotation. It consists of a single paragraph of text. There is no paragraph indentation.

and longer ones.

> This is a longer quotation. It consists of two paragraphs of text. The beginning of each paragraph is indicated by an extra indentation.
>
> This is the second paragarph of the quotation. It is just as dull as the first paragraph.

Another frequently-displayed structure is a list. The following is an example of an *itemized* list.

- This is the first item of an itemized list. Each item in the list is marked with a "tick". The document style determines what kind of tick mark is used.
- This is the second item of the list. It contains another list nested inside it. The inner list is an *enumerated* list.
  1. This is the first item of an enumerated list that is nested within the itemized list.
  2. This is the second item of the inner list. LaTeX allows you to nest lists deeper than you really should.

  This is the rest of the second item of the outer list. It is no more interesting than any other part of the item.
- This is the third item of the list.

You can even display poetry.

> There is an environment for verse
> Whose features some poets will curse.
>
> For instead of making
> Them do *all* line breaking,
> It allows them to put too many words on a line when they'd rather be forced to be terse.

Mathematical formulas may also be displayed. A displayed formula is one-line long; multiline formulas require special formatting instructions.

$$x' + y^2 = z_i^2$$

Don't start a paragraph with a displayed equation, nor make one a paragraph by itself.

## 10.4 Build process

To build LaTeX documents you need the latex program. It is free and available on all operating systems. Download and install. Many of us use the TexLive distribution and are very happy with it. You can use a editor and command line or use an IDE. To build this document via command line:

```
alta>  pdflatex SystemTemplate
```

If you change the bib entries, then you need to update the bib files:

```
alta>  pdflatex SystemTemplate
alta>  bibtex SystemTemplate
alta>  pdflatex SystemTemplate
alta>  pdflatex SystemTemplate
```

## Acknowledgement

Thanks to Leslie Lamport

# Bibliography


[1] R. Arkin. *Governing Lethal Behavior in Autonomous Robots*. Taylor & Francis, 2009.

[2] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George A Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, June 2005.

[3] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[4] V. Lumelsky and A. Stepanov. Path planning strategies for point mobile automation moving amidst unknown obstacles of arbirary shape. *Algorithmica*, pages 403–430, 1987.

[5] S.A. NOLFI and D.A. FLOREANO. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. A Bradford book. A BRADFORD BOOK/THE MIT PRESS, 2000.

[6] Wikipedia. Asimo — Wikipedia, the free encyclopedia. `http://upload.wikimedia.org/wikipedia/commons/thumb/0/05/HONDA_ASIMO.jpg/450px-HONDA_ASIMO.jpg`, 2013. [Online; accessed June 23, 2013].