Software Requirements Specification
for
Odyssey Space Research

Software Engineer Evaluation

Version 0.1

Prepared by **Frank Putnam, Jr.**

Date of submission: **Friday, June 7, 2019**

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
0.1	Frank Putnam, Jr.	Initial draft	2019/06/07

1 Introduction

1.1 Document Purpose

The purpose of this document is to outline software specifications of a C++ project to evaluate my software engineering abilities. The customer requirements were deliberately vague in order to review my personal decisions related to the implementation.

1.2 Product Scope

The product is a C++ class implementing a 2 dimensional matrix of form n x m, where n and m are set by the user when the object is constructed. The class will support operations of addition, scalar multiplication, matrix multiplication, and transpose (my choice). The overloaded C++ operators of ***get*** and ***set*** will manipulate data of a specific element in the matrix using unit-based access.
Unit tests will be created by me and included in the zipped git repository submission.

1.3 Intended Audience

The product will be evaluated by at least two senior software developers from Odyssey Space Research.

1.4 Definitions, Acronyms and Abbreviations

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.
TO DO: Please provide a list of all abbreviations and acronyms used in this document sorted in alphabetical order.>

SRS – Software Requirements Specifications
IEEE –
Class
Overloaded
Unit-based access
IDE

1.5 Document Conventions

In general this document follows the IEEE formatting requirements. Arial font size 11 or 12 is used throughout the document for text. *Italics* are used for comments and ***bold italics*** are used for code operators. Document text is single spaced and maintains 1” margins.

1.6 References and Acknowledgments

<*List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document.* >

1) edX: **Introduction to C++** (Provided by Microsoft)). Presented by Gerry O'Brien, Kate Gregory, and James McNellis.

2) Coursera On-line Education: **Matrix Algebra for Engineers** (The Hong Kong University of Science and Technology). Presented by Jeffrey R. Chasnov.

3) Wikipedia: *https://en.wikipedia.org/wiki/Matrix_multiplaction*

4) **A Firmware Development Standard** Version 1.4 *by* Jack G. Ganssle

2 Overall Description

2.1 Product Overview

<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. In this part, make sure to include a simple diagram that shows the major components of the overall system, subsystem interconnections, and external interface. In this section it is crucial that you will be creative and provide as much information as possible.

TO DO: Provide at least one paragraph describing product perspective. Provide a general diagram that will illustrate how your product interacts with the environment and in what context it is being used. This is not a formal diagram, but rather something that is used to illustrate the product at a high level.>

2.2 Product Functionality

<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 is needed here. These can be at the level given in the project description.>
TO DO: Provide a bulleted list of all the major functions of the system

1) C++ class implementing a 2 dimensional matrix of form is **n** x **m.**

2) **n** and **m** are:
   a. unsigned integers greater than or equal to 1.
   b. set by the user when the object is constructed.

3) Matrix element type is defined by the user at time of construction.

4) Operations of
   a. addition
   b. scalar multiplication
   c. matrix multiplication
   d. transpose

5) Overloaded operators
   a. ***get*** to obtain data from a specific element of the matrix.
   b. ***set*** to load data into a specific element of the matrix.
   c. Matrix element access is unit-based.

6) Constructors
   a. default initialization to **n** = 1_row and **m** = 1_column.
   b. custom initialization to **n** = Number_of_rows and **m** = Number_of_columns.

2.3 Design and Implementation Constraints

1) Use of standard C++ libraries.

2) IDE will be Microsoft Visual Studio 2019 Professional (trial version) with Visual C++ Redistributables.

2.4 Assumptions and Dependencies

The inherently subjective evaluation will factor the following items into the analysis:

a) Does the code compile?
b) Does it do what it should?
c) Does it pass all unit tests?
d) Are the unit tests sufficient?
e) How readable and maintainable is the code?
f) Is the code robust enough to handle any edge cases?
g) Were good habits exhibited that would be expected of a professional at your level of expertise?
h) Were C++ best practices followed (or failed to be followed)?
i) What extra features to these bare requirements were added?
j) Is the code efficient?
k) What design and implementation decisions were made, and why?
l) Were relevant comments and other documentation added?

2.5 External Interface Requirements

2.5.1 User Interfaces

The matrix header file **'OSR_matrix.h'** is included in the development environment.

2.5.2 Hardware Interfaces

Development and testing was performed on a Hewlett Packard Pavilion series laptop containing
a) x64 processor - AMD A8-6410 APU (4 CPUs) with AMD Radeon R5 graphics at 2.00 GHz.
b) DRAM - 16.0 GB.
c) Hard drive - Samsung 860 EVO SSD 1.00 TB.
d) Microsoft Windows 64-bit operating system.
    1) Windows 10 Home edition
    2) Version 1809
    3) OS build 17763.503

The C++ matrix class should run on any hardware platform as long as it is compiled for that platform.

2.5.3 Software Interfaces

For the default constructor:

```
OSR_matrix New_default_matrix{}; // matrix of bytes with 1 row and 1 column.
```

For the custom constructor:

```
OSR_matrix New_custom_matrix{n, m, element_type};
// matrix of element_type with n rows and m columns.
```

For the overloaded operator ***get*** a unit-based row number and unit-based column number are input and the value contained in the specified element of the matrix is returned.

For the overloaded operator ***set*** a unit-based row number, unit-based column number and the value to be placed in the specified element of the matrix are input.

For the operations of addition, and matrix multiplication two matrices are input and one matrix is returned.

For the operation of scalar multiplication one matrix and a scalar value are input and one matrix is returned.

For the operation of transpose one matrix is input and one matrix is returned.

2.6 Functional Requirements

2.6.1 **FR1:** The matrix class shall have a default constructor of 1 row and 1column of bytes.

2.6.2 **FR2:** The matrix class shall have a custom constructor of n rows and m columns of element_type.

2.6.3 **FR3:** The matrix element_type shall be any one of the following 4 types declared by the user at instantiation. BYTE_TYPE, FLOAT_TYPE, INT_TYPE, and UINT_TYPE.

2.6.4 **FR4:** The matrix elements shall be accessed via unit-based addressing. First element is accessed by (1,1). Last element accessed by (n, m).

2.6.5 **FR5:** The overloaded operator ***get*** shall obtain data from a specific element of the matrix. Element_contents = get_from(Row, Column);

2.6.6 **FR6:** The overloaded operator ***set*** shall load data into a specific element of the matrix. set(Row, Column, With_value);

2.6.7 **FR7:** The class shall support the operation of addition. Given two matrices **A** and **B** create a third matrix **C** with the elements of **A** and **B** added together.

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} e & f \\ g & h \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} a+e & b+f \\ c+g & d+h \end{bmatrix}$$

2.6.8 **FR8:** The class shall support the operation of scalar multiplication. Given the matrix **A** and a scalar value **K** create a second matrix **B** with the elements of **A** multiplied by the scalar **K**.

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \textbf{scalar} = k \quad \mathbf{B} = \begin{bmatrix} ka & kb \\ kc & kd \end{bmatrix}$$

2.6.9 **FR9:** The class shall support the operation of matrix multiplication. Given two matrices **A** and **B** create a third matrix **C** with the elements of **A** row multiplied by the elements of **B** column and then added together.

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} e & f \\ g & h \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{bmatrix}$$

2.6.10 FR10: The class shall support the operation of transpose. Given a matrix **A** create a second matrix $\mathbf{A}^T$ whose elements are the transpose of **A.**

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & *** & a_{1m} \\ a_{21} & a_{22} & *** & a_{2m} \\ * & * & *** & * \\ * & * & *** & * \\ a_{n1} & a_{n2} & *** & a_{nm} \end{pmatrix} \qquad \mathbf{A}^T = \begin{pmatrix} a_{11} & a_{21} & *** & a_{n1} \\ a_{12} & a_{22} & *** & a_{n2} \\ * & * & *** & * \\ * & * & *** & * \\ a_{1m} & a_{2m} & *** & a_{nm} \end{pmatrix}$$

3 Other Non-functional Requirements

3.1 Performance Requirements

<*If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.*
TODO: Provide performance requirements based on the information you collected from the client/professor. For example, you can say "P1. The secondary heater will be engaged if the desired temperature is not reached within 10 seconds">

3.2 Safety and Security Requirements

<*Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied. Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements.*
TODO:

- *Provide safety/security requirements based on your interview with the client - again you may need to be somewhat creative here. At the least, you should have some security for the mobile connection.*

3.3 Software Quality Attributes

<*Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.*

TODO: Use subsections (e.g., 4.3.1 Reliability, 4.3.2 Adaptability, etc…) provide requirements related to the different software quality attributes. Base the information you include in these subsections on the material you have learned in the class. Make sure, that you do not just write "This software shall be maintainable…" Indicate how you plan to achieve it, & etc…Do not forget to include such attributes as the design for change (e.g. adapting for different sensors and heating/AC units, etc.). Please note that you need to include ***at least*** *2 quality attributes. You can Google for examples that may pertain to your system.*>

4 Other Requirements

*<This section is **Optional**. Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>*

Appendix A – Data Dictionary

<Data dictionary is used to track all the different variables, states and functional requirements that you described in your document. Make sure to include the complete list of all constants, state variables (and their possible states), inputs and outputs in a table. In the table, include the description of these items as well as all related operations and requirements.>

Appendix B – Peer Review Log

<Please include here all the minutes from your group meetings, your group activities, and any other relevant information that will assist in determining the effort put forth to produce this document>