# Change History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>February 3, 2003</td>
<td>***************</td>
<td>First submission to team</td>
</tr>
<tr>
<td>1.1</td>
<td>February 14, 2003</td>
<td>***************</td>
<td>Grammatical changes, other misc. changes</td>
</tr>
</tbody>
</table>
# Table of Contents

1. **INTRODUCTION**  
   1.1 PURPOSE OF THE DOCUMENT  
   1.2 SCOPE OF THE PROJECT  
   1.3 DEFINITIONS, ACRONYMS AND ABBREVIATIONS  
   1.4 REFERENCES  
   1.5 OVERVIEW OF THE DOCUMENT  

2. **GENERAL DESCRIPTION**  
   2.1 PRODUCT FUNCTIONS  
   2.2 USER CHARACTERISTICS  
   2.3 USER OBJECTIVES  
   2.4 GENERAL CONSTRAINTS  
   2.5 NON-GOALS  

3. **FUNCTIONAL REQUIREMENTS**  
   3.1 USER INTERFACE  
      3.1.1 Description  
      3.1.2 Input  
      3.1.3 Output  
      3.1.4 Criticality  
      3.1.5 Technical Issues  
      3.1.6 Open Issues  
      3.1.7 Cost and Schedule  
      3.1.8 Risks  
      3.1.9 Dependencies with other requirements  
   3.2 TRIANGLE VALIDITY  
      3.2.1 Description  
      3.2.2 Input  
      3.2.3 Output  
      3.2.4 Criticality  
      3.2.5 Technical Issues  
      3.2.6 Open Issues  
      3.2.7 Cost and Schedule  
      3.2.8 Risks  
   3.3 TRIANGLE VALIDITY II  
      3.3.1 Description  
      3.3.2 Input  
      3.3.3 Output  
      3.3.4 Criticality  
      3.3.5 Technical Issues  
      3.3.6 Open Issues  
      3.3.7 Cost and Schedule  
      3.3.8 Risks  
      3.3.9 Dependencies with other requirements
1. Introduction

1.1 Purpose of the Document
This document is intended to describe the required functions of a program that is to accept user input, validate that the input describes a triangle, classify the triangle as isosceles, equilateral, or scalene, and draw the triangle. The software testers, programmers and design team will need to make use of this document in order to properly design, test, code, and maintain the suite.

1.2 Scope of the Project
This program will be intended to meet the demand of all users who desire to know the classification of a particular triangle, and exactly what it will look like when the triangle is drawn on the screen.

1.3 Definitions, Acronyms and Abbreviations

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinate set</td>
<td>(&lt;x, y&gt;) describes a single endpoint.</td>
</tr>
<tr>
<td>Coordinate pair</td>
<td>(&lt;x_1, y_1&gt;, &lt;x_2, y_2&gt;) describes both endpoints and thus a line.</td>
</tr>
<tr>
<td>Endpoint Connection</td>
<td>Point in space where two lines come together.</td>
</tr>
</tbody>
</table>

Ex 1. Given line A with endpoints \(<x, y>\) and \(<x_1, y_1>\) and line B with endpoints \(<x, y>\) and \(<x_2, y_2>\).

*Endpoint Connection* occurs at \(<x, y>\)

Common Endpoint: Same as *Endpoint Connection*

Uncommon Endpoint: Endpoints of a pair of lines which are not equivalent

Ex 2. Given line A with endpoints \(<x, y>\) and \(<x_1, y_1>\) and line B with endpoints \(<x, y>\) and \(<x_2, y_2>\).

*Uncommon Endpoints* occur at \(<x_1, y_1>\) and \(<x_2, y_2>\)
1.4 References

1.5 Overview of the document
This document contains several sections: Section 2 provides the General description of the program and Section 3 provides the functional requirements of the program.

2. General Description

2.1 Product Functions
The program will accept user-entered input to describe the characteristics of a set of three lines. The set of lines will then be tested to conform to the properties of a triangle. If the set does form a valid triangle, then the triangle will be classified as isosceles, equilateral, or scalene based on the relationships between the lengths of the sides. The resulting valid triangle should then be rendered on the screen.

2.2 User Characteristics
It should not be assumed that the users of the program have extensive knowledge of the system. It can be assumed, however that the users will be competent in entering data to describe the sets of lines. In the event of an error, the user should be advised of the error and how to correct the omission/mistake.

2.3 User Objectives
The user shall use this program to input different endpoints or lengths for three lines with the intention of the program rendering the resulting triangle. This shall be of particular use to a user who desires to know what the triangle will look like visually, given sets of endpoints or sets of line lengths.

2.4 General Constraints
This program is to run on a Windows platform, coded in Java.

2.5 Non-Goals
A simple, basic user interface, with little complexity/sophistication beyond the minimum functional requirements as defined in section 2.1.
3. Functional Requirements

3.1 User Interface

3.1.1 Description
The User shall enter points/lengths through a graphical interface. The data two describe the lengths will be entered in either coordinate <x,y> form, or line length. The user shall specify the method before entering points. The user shall enter three coordinate pairs, or three line lengths. User shall be prompted for a complete coordinate (both the X and Y component), or line length of one line before entering values to describe the next line.

3.1.2 Input
None

3.1.3 Output
Will propagate the sets of coordinates through to the next steps of the program. This will most likely be to test the coordinates for validity.

3.1.4 Criticality
The GUI is not as critical as the other aspects of the program.

3.1.5 Technical Issues
Proper GUI techniques should be researched and implemented. User requirements of the users should be clearly accessible and defined. In several functions, there will be output that will need to be printed for the user (see sections 3.2.3, 3.3.3).

3.1.6 Open Issues
Proper coding and implementation techniques for GUI should be researched.

3.1.7 Cost and Schedule
Implementation should be the final aspect of the program.

3.1.8 Risks
It will be necessary to research techniques for the implementation. In the event that a GUI cannot be implemented, the interface shall be through a command line shell.

3.1.9 Dependencies with other requirements
The user interface will supply other functions with the user-specified values for the endpoints/lengths of the lines.
3.2 Triangle Validity

3.2.1 Description

Given that the user has entered numeric values to describe the lengths of the respective lines, the program must run tests to ensure that the set of three lines is valid with respect to the definition of a triangle. A valid triangle should conform to the following standards:

- Given that the user has entered three numeric values to represent the lengths of the sides:
  1. The numeric values should be positive numbers
  2. Given lines A and B, with a single common endpoint, the length of the third side, line C, shall be equal to the distance between the uncommon endpoints of lines A and B.
- In the event that a line length is not equal to the distance between the uncommon endpoints of the other two lines, the triangle shall be judged invalid.
- The validity test shall allow for all permutations of sides A, B, C.

If the above description is satisfied, then the triangle is valid, and other tests shall be run. If one of the conditions is untrue, then the proper error report shall be generated and the program aborted. In the event that the triangle is invalid, the user will be advised and prompted for re-entry of line lengths.

3.2.2 Input

User will enter one numeric value that will correspond to the length of a side. The user should enter three such values, one for each line.

3.2.3 Output

Output shall include the acceptance or rejection of the set of lengths as a valid triangle. A valid return value will be generated upon acceptance of the group of line lengths. An appropriate error message printed upon rejection of the group of line lengths.

3.2.4 Criticality

This component is highly critical to the system, as it will be impossible to properly characterize and perform other aspects of the program on triangles that do not fit the definition of a triangle.
3.2.5 Technical Issues

There must be a structure to maintain the coordinate pairs of both endpoints associated with each endpoint. In addition, there must also be a method of checking that the endpoint connections are all different.

3.2.6 Open Issues

None

3.2.7 Cost and Schedule

This is a highly critical component, and should be tested extensively.

3.2.8 Risks

This is part of the basic functionality of the program, and should be satisfied under all circumstances.

3.3 Triangle Validity II

3.3.1 Description

Given that the user has entered the points to describe three line segments, the program must run a series of tests to test the validity of the endpoints with respect to the definition of a triangle. A triangle is, for the purpose of this program, defined as a figure enclosed by three line segments. The enclosure of the figure shall be described as:

- Given sides A, B, C, each with two endpoints described as \(<x_1, y_1>\) and \(<x_2, y_2>\)
  1. The endpoint \(<x_1, y_1>\) of line segment A shall be the same as \(<x_1,y_1>\) of line segment B.
  2. The endpoint \(<x_2, y_2>\) of line segment B shall be the same as \(<x_1, y_1>\) of line segment C.
  3. The endpoint \(<x_2, y_2>\) of line segment C shall be the same as \(<x_2, y_2>\) of line segment A.
  4. The lines A, B, C shall not all have a common endpoint.
  5. No two endpoints of a line shall be the same point
     - Given A with endpoints \(<x_1, y_1>, <x_2, y_2>\):
       \(<x_1, y_1> \neq <x_2, y_2>\)

The validity test shall allow for all permutations of sides A, B, C.

If the above description is satisfied, then the triangle is valid, and other tests shall be run. If one of the conditions is untrue, then the proper error
report shall be generated and the program aborted. In the event that the triangle is invalid, the user will be advised and prompted for re-entry of coordinate pairs.

3.3.2 Input

User will enter two endpoints per line. There shall be three lines, and thus the user will input six points. The points will be in a coordinate system <x, y>, with each coordinate a positive value.

3.3.3 Output

Output shall include the acceptance or rejection of the set of lengths as a valid triangle. A valid return value will be generated upon acceptance of the group of line lengths. An appropriate error message printed upon rejection of the group of line lengths.

3.3.4 Criticality

This component is highly critical to the system, as it will be impossible to properly characterize and perform other aspects of the program on triangles that do not fit the definition of a triangle.

3.3.5 Technical Issues

There must be a structure to maintain the coordinate pairs of both endpoints associated with each endpoint. In addition, there must also be a method of checking that the endpoint connections are all different.

3.3.6 Open Issues

None.

3.3.7 Cost and Schedule

This is a highly critical component, and should be tested extensively.

3.3.8 Risks

This is part of the basic functionality of the program, and should be satisfied under all circumstances.

3.3.9 Dependencies with other requirements

This is the basic functional requirement. This requirement will determine whether other functions of the program are permitted to operate on the user-entered data, or whether the data will be returned to the user.

3.4 Triangle Type

3.4.1 Description

Given that the Triangle is proven Valid, the sides should be tested to determine the specific type of triangle. The triangle types include Isosceles (section 3.5), Scalene (section 3.6), and Equilateral (section 3.7).
3.4.2 Input
User entered input as described in section 3.1.2.

3.4.3 Output
The triangle type shall be displayed based on the information supplied from the tests to determine the type of triangle. Upon successful determining the type of triangle, see below, the type shall be printed to the screen.

- If the triangle is determined to be an isosceles, “Isosceles Triangle” shall be printed.
- If the triangle is determined to be a scalene, “Scalene Triangle” shall be printed.
- If the triangle is determined to be an equilateral, “Equilateral Triangle” shall be printed.

3.4.4 Criticality
See sections 3.5, 3.6, 3.7

3.4.5 Technical Issues
See sections 3.5, 3.6, 3.7

3.4.6 Open Issues
None.

3.4.7 Cost and Schedule
See sections 3.4, 3.5, and 3.6.

3.4.8 Risks
This requirement will be dependent on the correct operation of the Isosceles, Scalene, and Equilateral Triangle functions (sections 3.5, 3.6, 3.7). In the event that any of the above do not operate properly, this functionality will have compromised applicability.

3.4.9 Dependencies with other requirements
This requirement will be dependent on the correct operation of the Isosceles, Scalene, and Equilateral Triangle functions (sections 3.3, 3.4, 3.5).

3.5 Isosceles Classification

3.5.1 Description
Given that the Triangle is proven Valid, the sides should be tested to determine whether two, and only two, of the three sides are of equal length. This is to be a sub-function of the Triangle Type function (section 3.4).
3.5.2 Input
This requirement/function will be supplied with data from the Triangle Type (section 3.4) requirement.

3.5.3 Output
This sub-function should provide a flag to the parent function to denote success/failure that the input conforms to the terms of this type of triangle.

3.5.4 Criticality
This functionality is critical to the correct operation of the Triangle Type (section 3.4) function.

3.5.5 Technical Issues
This is a sub function of the Triangle Type (section 3.4) function.

3.5.6 Open Issues
None

3.5.7 Cost and Schedule
Should be completed fairly early in the implementation and testing process.

3.5.8 Risks
None

3.6 Scalene Classification

3.6.1 Description
Given that the triangle in question is valid, it should be tested to determine if all three sides are unequal. If this test is true, then the triangle is determined to have the properties of a scalene Triangle. This is to be a sub-function of the Triangle Type function (section 3.4).

3.6.2 Input
This requirement/function will be supplied with data from the Triangle Type (section 3.4) requirement.

3.6.3 Output
This sub-function should provide a flag to the parent function to denote success/failure that the input conforms to the terms of this type of triangle.

3.6.4 Criticality
This functionality is critical to the correct operation of the Triangle Type (section 3.4) function.

3.6.5 Technical Issues
This is a sub function of the Triangle Type (section 3.4) function.
3.6.6 Open Issues

None

3.6.7 Cost and Schedule

Should be completed fairly early in the implementation and testing process.

3.6.8 Risks

None

3.7 Equilateral Classification

3.7.1 Description

Given that the triangle in question is valid, it should be tested to determine if the three sides are equal in length. This is to be a sub-function of the Triangle Type function (section 3.4).

3.7.2 Input

This requirement/function will be supplied with data from the triangle type (section 3.4) requirement.

3.7.3 Output

This sub-function should provide a flag to the parent function to denote success/failure that the input conforms to the terms of this type of triangle.

3.7.4 Criticality

This functionality is critical to the correct operation of the Triangle Type (section 3.4) function.

3.7.5 Technical Issues

This is a sub function of the Triangle Type (section 3.4) function.

3.7.6 Open Issues

None

3.7.7 Cost and Schedule

Should be completed fairly early in the implementation and testing process.

3.7.8 Risks

None