StOut

MTM Program Product

Software Requirements Specification

*Version 0.2.1*

*9/11/2017*

Applying MTM SRS VERSION 4.0

**Project Director:** tbs

**Project Manager:** tbs

**Project Team:** tbs

**Document Author:** tbs

Version History

|  |  |  |  |
| --- | --- | --- | --- |
| *Version* | *Date* | *Authors* | *Comment* |
| 0.1 | 9/11/2017 | Jesse Anderson | Properties, Variables, Theming, and other formatting. |
| 0.1.1 | 9/13/2017 | Trevor Brooks | Added some more information from original SRS of AbOut. Also made comments for future updates. As a future note, all references to AbOut should be removed. |
| 0.2.1 | 11/5/2017 | Jesse Anderson, Trevor Brooks, Nathaniel Lewis, Abdulrahman E. Alduraiweesh | General Factors, Requirements, |

TABLE CONTENTS

[1 Introduction 1](#_Toc331683624)

[1.1 Software Purpose and Scope 1](#_Toc331683625)

[1.2 Document Purpose and Contents 1](#_Toc331683626)

[1.3 Definitions, Acronyms, and Abbreviations 1](#_Toc331683627)

[1.3.1 Definitions 1](#_Toc331683628)

[1.3.2 Acronyms and Abbreviations 3](#_Toc331683629)

[1.3.3 Technical Definitions/Data Dictionary 4](#_Toc331683630)

[1.4 References 4](#_Toc331683631)

[2 General Factors 4](#_Toc331683632)

[2.1 Product Perspective 5](#_Toc331683633)

[2.2 Product Functions 5](#_Toc331683634)

[2.3 Environmental Conditions 5](#_Toc331683635)

[2.4 User Characteristic 6](#_Toc331683636)

[2.5 Dependencies 7](#_Toc331683637)

[2.6 Assumptions 7](#_Toc331683638)

[3 Analysis Use Cases 7](#_Toc331683639)

[4 Explanatory User Interfaces 8](#_Toc331683640)

[5 Specific Requirements 8](#_Toc331683641)

[5.1 Functional Requirements 9](#_Toc331683642)

[5.2 Non-Functional Requirements 9](#_Toc331683643)

[5.2.1 Design Constraints (DC) 9](#_Toc331683644)

[5.2.2 Human Factors (HF) 9](#_Toc331683645)

[5.2.3 External Interface Requirements (XI) 9](#_Toc331683646)

[5.2.4 Security (SC) 10](#_Toc331683647)

[5.2.5 Development Environment (DV) 10](#_Toc331683648)

[5.2.6 Standards (ST) 10](#_Toc331683649)

[5.2.7 Delivery Environment (DL) 10](#_Toc331683650)

[5.2.8 Performance (PR) 10](#_Toc331683651)

[5.2.9 Deliverable Items, Dates and Conditions (DD) 10](#_Toc331683652)

[5.2.10 Cost (CT) 10](#_Toc331683653)

[5.2.11 Quality (QL) 10](#_Toc331683654)

[5.2.12 V&V Activities (VV) 11](#_Toc331683655)

[5.2.13 Database (DB) 11](#_Toc331683656)

[5.2.14 Adaptability 11](#_Toc331683657)

[5.3 Requirements Models 11](#_Toc331683658)

[6 Illustrative Use Cases (IUC) 11](#_Toc331683659)

[7 Future Enhancements (FE) 12](#_Toc331683660)

# Introduction

*[This Software Requirements Specification template is designed to facilitate the definition of processes and procedures relating to software requirements specification activities. This template was developed using IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications.*

Information displayed in brackets is explanatory. Delete the bracketed text items and add your project-specific input. These items are food for thought on the section they address.

*The introduction section should explain the purpose and scope of the project software requirements specification (SRS), as well as, provide clarification of definitions, acronyms, and references. This section should also provide an overview of this document.*

*Place any material here that is not specific to any of the sub-sections below.]*

## Software Purpose and Scope

The goal of this web application is to simplify and standardize how faculty members in the Computer Science (CS) and Electrical Engineering (EE) Departments at Montana Tech assess their courses in relation to ABET student outcomes. Specifically, help CS and EE faculty members determine the extent to which students in their courses have met student outcomes by streamlining the repetitive tasks which the faculty members were doing by hand. This application should make continual course assessment easier. The customers for this web application are the faculty and staff of the CS and EE department. The web application is called AbOut, for Accreditation Board of Engineering and Technology (ABET) Outcomes.

## Document Purpose and Contents

*[This subsection should explain the purpose for writing an SRS for this project and describe the intended audience for the SRS. This subsection should describe the information that will be presented in each of the subsections from §2 on.]*

## Definitions, Acronyms, and Abbreviations

The following tables of definitions, acronyms and references may be useful for reading this document.

### Definitions

|  |  |
| --- | --- |
| Administer | See “Authorized Administrator” |
| Authorized Administrator | Member in the MTECHS domain who is registered in the AbOut system and has been assigned the administrator role for one or more programs. |
| Authorized Faculty Member | Montana Tech faculty member in the MTECHS domain who is registered in the AbOut system and has been assigned to teach one or more course offerings. Faculty members may be “active” or “inactive”. |
| Authorized Observer | Anyone with an account in the MTECHS system who has been assigned the “observer” role. |
| Authorized User | Montana Tech faculty member or staff in the MTECHS domain who is registered in the AbOut system. In addition, anyone with an account in the MTECHS system who has been assigned the “observer” role. |
| CORE | Course Outcome, Review and Evaluation (CORE) |
| CORE Report | A report which faculty members in the CS Department are required to write for each course offering which they teaching in the CS or SE degree. The AbOut web application generates statistics for this report. (See CORE Statistics) |
| CORE Statistics | Report showing the extent to which student outcomes were met by students in a course offering. |
| Course | Course contributing to measuring student outcomes and offered by the CS department. Courses are identified by a prefix and number, such as ESOF 328. The course prefixes are CSCI and ESOF. The course has a name, such as Software Requirements and Specifications. |
| Course Offering | A particular section of a course offered in a particular semester. Course offerings are identified by a course, a section, and a semester. |
| Course Outcome | Criteria which students passing the course should meet. These are specific to the course and are different than ‘Student outcomes’ which are specific to either the CS or SE program. AbOut does not measure course outcomes. This term does not appear elsewhere in this document and is included here to avoid confusing this with student outcomes. |
| Course PC Report | Report showing the extent to which students met performance criteria during the semester(s) of interest. This report is divided by performance criteria and courses. |
| Default Semester | The semester used when no semester is given (for instance, creating a course offering or a new course). |
| Faculty Member | See “Authorized Faculty Member” |
| Matrix Report | Report showing the association of courses to performance criteria and the weights of those associations. This is used to get an overview of the extent to which courses are covering performance criteria. |
| Metric | A metric associated with a course offering. The metric is created by the instructor of the course. It consists of a description, maximum number of points, and a list of the student outcomes which it measures. |
| Metric Goal | The overall (percentage) score which a student needs to meet or exceed to be considered to have met the student outcomes.  This needs to be stored in such a way so the metric goal can be changed easily. Throughout this document, it is assumed that the metric goal is 70%. |
| Modern Browser | Any web browser which reliably implements the latest (as of 2011) standards in HTML and CSS, with complete support for JavaScript. |
| Outcome Report | Report showing the extent to which students met the selected student outcome during the chosen semester. This report is divided by the performance criteria which measure the outcome. |
| PC Semester Report | Report showing the extent to which students met performance criteria during the semester(s) of interest. This report is divided by performance criteria and semesters. |
| Performance Criterion | A criterion which a program will use to assess a student outcome. |
| Performance Criterion Abbreviation | Abbreviation which will be used to identify an performance criterion. |
| Program | A degree program. This system  is to facilitate assessment for the Computer Science, Electrical Engineering and Software Engineering programs. |
| Program Abbreviation | Abbreviation which will be used to identify a program. |
| Raw data | Raw data is the number of students, weight of the course, and percentage. |
| Registered in AbOut | AbOut contains user information for this person. |
| Semester | A Montana Tech semester. Consists of a year and either fall, spring or summer. |
| Semester(s) of Interest | A single semester or a range of semesters for which a report is being generated. |
| Simple Average | An average of each value with no weighting due to characteristics of the values. |
| Student Score | The score that a student earned on a metric. |
| Student Outcome | A criterion which a program will assess for ABET accreditation. |
| Student Outcome Abbreviation | Abbreviation which will be used to identify a student outcome. |
| User | See “Authorized User” |

### Acronyms and Abbreviations

|  |  |
| --- | --- |
| AD | Adaptability |
| AL | Availability |
| AUC | Analysis Use cases |
| CM | Communications |
| CT | Cost |
| DB | Database |
| DC | Design Constraint |
| DD | Delivery Data and Conditions |
| DL | Delivery Environment |
| DV | Development Environment |
| EN | Enhanceability/Extendibility |
| FE | Future Enhancements |
| HF | Human Factors |
| HW | Hardware |
| IUC | Illustrative Use Cases |
| ML | Maintainability |
| OP | Operations |
| PR | Performance |
| PT | Portability |
| QL | Quality |
| RL | Reliability |
| SC | Security |
| SDD | Software Design Description |
| SRS | Software Requirements Specification |
| ST | Standards |
| SI | Site |
| SW | Software |
| UB | Usability |
| VV | Verification & Validation |
| XI | External Interfaces |
| XXX | X of X of X |

### Technical Definitions/Data Dictionary

*[This subsection shall list (alphabetically) and briefly describe all data collections and items mentioned in this SRS. If a data base is not involved the technical items names in the requirements are defined here. These names should be chosen with care. The expectation is that these names will be used later in the design and implementation.]*

|  |  |  |
| --- | --- | --- |
| ItemName | Type | Brief description of data item |
| First item | Table | A description |

## References

ABET, <http://www.abet.org/>

CAS, <https://wiki.jasig.org/display/CAS/Home>

CS Department Student Outcomes, <http://cs.mtech.edu/main/index.php/component/content/article/146>

W3C XHTML validation software, [http://validator.w3.org](http://validator.w3.org/)

W3C CSS validation software, <http://jigsaw.w3.org/css-validator>

# General Factors

*[The General Factors section should describe the general factors that affect the product and its requirements. Place any material here that is not specific to any of the sub-sections.*

*In this and each subsequent major section, briefly describe the purpose of this section from the readers perspective.]*

## Product Perspective

This web application will be independent of other products except CAS (see Section 2.5, Dependencies).

## Product Functions

This section provides a high-level overview of the functionality of the web application.

#### Function Overview

The AbOut web application will be used to:

* Record student outcomes associated with the computer science, electrical engineering and software engineering programs
* Associate performance criteria with student outcomes
* Associate performance criteria with courses
* Associate faculty and students with course offerings
* Enable faculty to record metrics of the course offerings they teach
* Enable faculty to record the scores which a student earned on an metric
* Generate a variety of reports indicating the extent to which performance criteria were met
* Allow observers to see reports and all of the information leading to the report, with student names redacted

These functions are divided into three overlapping sets: faculty, administrative, and reporting functions.

#### Administrative Functions

Authorized faculty members and administrators are able to do the following:

* Add, edit, delete and view users of the AbOut system
* View the semesters in the system and set a default semester
* Tell the system to generate the next chronological semester
* Add, edit, delete and view student outcomes
* Add, edit, delete and view performance criteria for student outcomes
* Add, edit, delete and view courses, along with the performance criteria associated with them
* Add, edit, delete and view offerings of courses
* Add, edit, delete and view students in a course offering
* Import a list of students into a course.

Note that administrators are not able to add, edit or delete metrics associated with a course offering, and that when administrators view student scores on a metric, student names will be redacted.

If a user is both a faculty member and an administrative, that user will be able to view and edit the information associated with a course offering which they teach.

#### Faculty Functions

Authorized faculty members are able to do the following:

* View the course offerings that they are currently teaching or have taught in the past
* Add metrics to course offerings that they have taught or are teaching
* Add or remove students from course offerings that they taught or are teaching
* Enter student scores on the metrics in the course offering which they taught or are teaching
* Export a list of the students in an offering they have taught or are teaching.

#### Reporting Functions

Authorized faculty members and administrators are able to do the following.

* Generate CORE statistics showing the extent to which students enrolled in a course offering met the performance criteria associated with that offering
* Generate a Course PC Report showing the extent to which students met performance criteria per course during the semester(s) of interest.
* Generate a Matrix Report showing the weights associated with courses assessing performance criteria for a specified program. Note that this report generates the information for the current semester.
* Generate an Outcome Report showing the extent to which students met performance criteria for a semester.
* Generate a Course PC Semester Report showing the extent to which students met performance criteria per semester over the semester(s) of interest.

## Environmental Conditions

AbOut will be a web application which is accessed from the CS Department website. The CS Department will need a web and database server in order to serve AbOut.

Users will need a Montana Tech account in the MTECHS domain to access this system. They will also need to be registered within the AbOut system, i.e. AbOut needs to contain the user id for this person.

## User Characteristic

The primary users of this system are the faculty and staff of the CS Department. Someone who has a username in the MTECHS, is registered in AbOut and has been assigned the role of “observer” for one or more programs will also be a user of this system. An understanding of the assessment process, a familiarity with web browsers, and proficiency completing forms on a computer, is assumed.

Users can be assigned one or more of the roles: administrator, faculty, and observer.

## Dependencies

AbOut will utilize the Central Authentication Service (CAS). CAS is a single sign-on protocol for the web. CAS allows web applications to authenticate users without gaining access to a user’s security credentials.

CAS is offered by Montana Tech’s Campus Technology Services and is used to authenticate Montana Tech users for most campus applications. Without CAS, users of AbOut would need to create and remember another username / password combination to login to AbOut.

## Assumptions

*[This subsection should list all assumptions that on which the software resulting from the SRS will depend that have not been covered above. This subsection should be the source for recognizing the impact of any changes to these assumptions on the SRS and resulting software.. This section can highlight unresolved requirement issues that should be recorded on the Project Manager’s Open Issues List.]*

# Analysis Use Cases

## Actors

|  |  |  |
| --- | --- | --- |
| Primary Actor | Description | Use Cases |
| Administrator | A permission that allows users to do administrative functions for assessments. This permission is meant for administrative staff.    Examples:  Secretaries and Department heads | Add/edit/delete/view student courses  Add/edit/delete/view student outcomes  Add performance criteria to student outcome  Edit/delete/view performance criteria  Add/edit/delete/view users  Add/edit/delete/view course offerings  Generate next semester  Import lists of students  Set default semester  View semesters in system |
| Observer | A permission that allows the user to view data, with student names redacted. These users are not allowed to modify data. | View student outcomes  View courses  View course offerings  View metrics associated with a course offering  Generate C.O.R.E. Report  Generate Course PC Report  Generate Matrix Report  Generate Outcome Report  View data of a class, with student names redacted. |
| Faculty member active | A permission that allows the user limited add, edit, view, and delete permissions to offerings they are associated with. The user is also allowed to generate C.O.R.E statics. | Add/remove students from course offering  Add/edit/delete/view metric within course offering  Add score(s) to metric  Export list of students  View course offering  Generate C.O.R.E. statistics |
| Faculty member inactive | A permission for ex faculty members. Meant to help preserve data for historical reference. | None |

## Use Cases

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name: | Get Course Reporting Report (who has / hasn’t input) | | |
| Created By: | Trevor Brooks | Last Updated By: | Trevor Brooks |
| Date Created: | 10-9-17 | Date Last Updated: |  |

|  |  |
| --- | --- |
| Actors: | Program Coordinator |
| Description: | Program coordinators may wish to lookup which instructors / courses are missing data for this semester. |
| Trigger: | End of semester approaching. |
| Preconditions: | 1. Courses set up in the system. |
| Post conditions: | 1. Report has been generated. |
| Normal Flow: | * 1. Coordinator goes to the reporting section.   2. Coordinator clicks ‘Generate Reporting Report’   3. System displays report |
| Alternative Flows: |  |
| Exceptions: | * + - * 1. No courses listed for this semester. |
| Includes: | None |
| Priority: | High |
| Frequency of Use: | Each semester |
| Business Rules: |  |
| Special Requirements: | None |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name: | Set input deadline | | |
| Created By: | Trevor Brooks | Last Updated By: | Trevor Brooks |
| Date Created: | 10-9-17 | Date Last Updated: |  |

|  |  |
| --- | --- |
| Actors: | Program Coordinator |
| Description: | Some programs would like to set a deadline for input of data. This allows the coordinator to set this deadline. |
| Trigger: |  |
| Preconditions: | 1. Program Coordinator is set. 2. Program has courses set up for this semester. |
| Post conditions: | 1. Deadline is set for a program. |
| Normal Flow: | * 1. Coordinator goes to program management view.   2. Program coordinator sets a date in the future, after the beginning of a semester.   3. Coordinator confirms selection. |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: | None |
| Priority: | High |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: | None |
| Assumptions: |  |
| Notes and Issues: | * This could cause the rollover of operational to analytical data. |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name: | Get reports (analytical data) | | |
| Created By: | Trevor Brooks | Last Updated By: | Trevor Brooks |
| Date Created: | 10-9-17 | Date Last Updated: |  |

|  |  |
| --- | --- |
| Actors: | Program Coordinator |
| Description: | Coordinator would like to see the data collected to prepare accreditation reports. |
| Trigger: | Coordinator needs data for preparing reports or presentations. |
| Preconditions: |  |
| Post conditions: |  |
| Normal Flow: | * 1. Coordinator goes to the reporting section.   2. Coordinator selects a report.   3. Coordinator supplies the needed data (semesters, program, etc).   4. System generates the report. |
| Alternative Flows: |  |
| Exceptions: | * + - * 1. Needed data not available in specified range. |
| Includes: | None |
| Priority: | High |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: | None |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name: | View operational (raw) data | | |
| Created By: | Trevor Brooks | Last Updated By: | Trevor Brooks |
| Date Created: | 10-9-17 | Date Last Updated: |  |

|  |  |
| --- | --- |
| Actors: | Program Coordinator |
| Description: | Coordinators need to see a view of data that is being entered to see current status. |
| Trigger: | Coordinator has a question about a course. |
| Preconditions: |  |
| Post conditions: |  |
| Normal Flow: | * 1. Coordinator goes in to the view containing active data.   2. Coordinator selects a course.   3. System displays current data entered, as well as all tied data.   4. Coordinator can edit the displayed data and save by clicking submit. |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: | None |
| Priority: | High |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: | None |
| Assumptions: | * Course in question is set up for this semester. |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name: | Set program configuration | | |
| Created By: | Trevor Brooks | Last Updated By: | Trevor Brooks |
| Date Created: | 10-9-17 | Date Last Updated: |  |

|  |  |
| --- | --- |
| Actors: | Program Coordinator |
| Description: | Coordinators can set various settings such as deadlines and whether or not to track student names in operational data. |
| Trigger: |  |
| Preconditions: | 1. Program is set up. |
| Post conditions: |  |
| Normal Flow: | * 1. Coordinator goes to program management view.   2. Coordinator is shown the current setup for the program.   3. Coordinator can set new settings.   4. Any new settings are saved by clicking ‘submit’. |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: | None |
| Priority: | High |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: | None |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name: | Add course | | |
| Created By: | Trevor Brooks | Last Updated By: | Trevor Brooks |
| Date Created: | 10-9-17 | Date Last Updated: |  |

|  |  |
| --- | --- |
| Actors: | Program Coordinator |
| Description: | Courses must be set up, and new courses need to be tracked. |
| Trigger: | Program was just set up or a new course was created in a program. |
| Preconditions: | 1. Program exists. |
| Post conditions: |  |
| Normal Flow: | * 1. Coordinator goes to program management view.   2. Coordinator selects “Add new course” from the actions menu.   3. Coordinator enters data for the course.   4. Submit button is clicked. |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: | None |
| Priority: | High |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: | None |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name: | Remove Course | | |
| Created By: | Trevor Brooks | Last Updated By: | Trevor Brooks |
| Date Created: | 10-9-17 | Date Last Updated: |  |

|  |  |
| --- | --- |
| Actors: | Program Coordinator |
| Description: | Program coordinator needs to remove a course that was added by mistake. |
| Trigger: |  |
| Preconditions: | 1. Course does not have any operational data other than instructor tied to it. (Course scores) |
| Post conditions: | 1. Course does not exist within operational part of the system. 2. List of outcomes applicable to the course is removed. |
| Normal Flow: | * 1. Coordinator goes to Program Management view.   2. Coordinator selects the course they are wanting to remove   3. Course data is displayed.   4. Coordinator clicks the Remove Course button.   5. A warning popup is displayed.   6. Coordinator accepts data loss. |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: | None |
| Priority: | High |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: | None |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name: | Update Course (general info, outcomes) | | |
| Created By: | Trevor Brooks | Last Updated By: | Trevor Brooks |
| Date Created: | 10-9-17 | Date Last Updated: |  |

|  |  |
| --- | --- |
| Actors: | Program Coordinator |
| Description: | Course info may change over time. Coordinator should be able to edit this as operational data. |
| Trigger: |  |
| Preconditions: | 1. Course exists in the program. |
| Post conditions: |  |
| Normal Flow: | * 1. Coordinator goes to Program Administration view.   2. Coordinator selects course.   3. Coordinator changes relevant info.   4. Coordinator selects ‘Submit’. |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: | None |
| Priority: | High |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: | None |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name: | Add Program Outcome | | |
| Created By: | Trevor Brooks | Last Updated By: | Trevor Brooks |
| Date Created: | 10-9-17 | Date Last Updated: |  |

|  |  |
| --- | --- |
| Actors: | Program Coordinator |
| Description: | An outcome is tied to an assessment. Coordinators need to add these to the system. |
| Trigger: |  |
| Preconditions: | 1. Program exists. 2. Assessment exists. |
| Post conditions: | 1. Outcome added to assessment. |
| Normal Flow: | * 1. Coordinator goes to Program Administration view.   2. Coordinator selects assessment to add outcome to.   3. Coordinator selects ‘New outcome’   4. Outcome information is input.   5. ‘Submit’ is clicked. |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: | None |
| Priority: | High |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: | None |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name: | Remove Program Outcome | | |
| Created By: | Trevor Brooks | Last Updated By: | Trevor Brooks |
| Date Created: | 10-9-17 | Date Last Updated: |  |

|  |  |
| --- | --- |
| Actors: | Program Coordinator |
| Description: | Outcomes incorrectly added must be removed. |
| Trigger: |  |
| Preconditions: | 1. Outcome must not be tied to any courses. |
| Post conditions: |  |
| Normal Flow: |  |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: | None |
| Priority: | High |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: | None |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name: | Update Program Outcome | | |
| Created By: | Trevor Brooks | Last Updated By: | Trevor Brooks |
| Date Created: | 10-9-17 | Date Last Updated: |  |

|  |  |
| --- | --- |
| Actors: | Program Coordinator |
| Description: | Outcomes can change in either wording or in name between assessment cycles. |
| Trigger: |  |
| Preconditions: |  |
| Post conditions: |  |
| Normal Flow: |  |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: | None |
| Priority: | High |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: | None |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name: | Add Program Assessment | | |
| Created By: | Trevor Brooks | Last Updated By: | Trevor Brooks |
| Date Created: | 10-9-17 | Date Last Updated: |  |

|  |  |
| --- | --- |
| Actors: | Program Coordinator |
| Description: | Outcomes are part of assessments. Assessments such as ABET or Northwest must be added to each program. |
| Trigger: |  |
| Preconditions: | 1. Program exists in the system. |
| Post conditions: |  |
| Normal Flow: | * 1. Coordinator goes to Program Management view.   2. Coordinator selects “Add new assessment” from actions list.   3. Name of the assessment, and an optional description is supplied.   4. Coordinator clicks submit. |
| Alternative Flows: | **1.1**  **1.2** |
| Exceptions: |  |
| Includes: | None |
| Priority: | High |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: | None |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name: | Remove Program Assessment | | |
| Created By: | Trevor Brooks | Last Updated By: | Trevor Brooks |
| Date Created: | 10-9-17 | Date Last Updated: |  |

|  |  |
| --- | --- |
| Actors: | Program Coordinator |
| Description: | An assessment that has no data should be able to be removed if it is not important. |
| Trigger: |  |
| Preconditions: | 1. Program exists. 2. Assessment exists. 3. Assessment has no data tied to it. (Operational or analytical) |
| Post conditions: |  |
| Normal Flow: | * 1. Coordinator goes to the Program Management view.   2. Coordinator selects the assessment that should be deleted.   3. Coordinator clicks delete and confirms on the pop-up. |
| Alternative Flows: |  |
| Exceptions: | * + - * 1. Assessment has information associated with it. |
| Includes: | None |
| Priority: | High |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: | None |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name: | Update Program Assessment | | |
| Created By: | Trevor Brooks | Last Updated By: | Trevor Brooks |
| Date Created: | 10-9-17 | Date Last Updated: |  |

|  |  |
| --- | --- |
| Actors: | Program Coordinator |
| Description: | In case of a typo in the name or description of an assessment, it should be editable. |
| Trigger: |  |
| Preconditions: | 1. Assessment exists within program. |
| Post conditions: |  |
| Normal Flow: | * 1. Coordinator goes to program management view.   2. Coordinator selects the assessment to be modified.   3. Relevant information is edited.   4. User clicks submit. |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: | None |
| Priority: | High |
| Frequency of Use: |  |
| Business Rules: |  |
| Special Requirements: | None |
| Assumptions: |  |
| Notes and Issues: | * This could cause issues between operational / analytical data. Take care with structure. |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name: | Add Student Score(s) to Metric | | |
| Created By: | AbdulRahman Alduraiweesh | Last Updated By: | AbdulRahman Alduraiweesh |
| Date Created: | 10/9/2017 | Date Last Updated: | 10/9/2017 |

|  |  |
| --- | --- |
| Actors: | Faculty member |
| Description: | A faculty member enters a student score, or multiple students’ scores, to a metric. |
| Trigger: | Faculty member has created a metric and wants to enter student scores for the metric |
| Preconditions: | 1. Faculty member is within a course offering which (s)he is teaching and has created a metric for the offering by entering a description of the metric, its maximum number of points and at least one outcome which it measures 2. There is at least one student in the course offering |
| Post conditions: | 1. The score(s) entered are stored in StOut |
| Normal Flow: | 1. Enter a single score.    1. Faculty member inputs the number of points a student earned on the metric.    2. Faculty indicates “save”    3. The percentage(s) for the score(s) have been updated    4. A message informing the user that the scores were saved is displayed |
| Alternative Flows: | 1. Faculty member enters multiple scores (branch before step )    1. Faculty member has a list of newline separated values in the clipboard    2. Faculty member inputs these scores (return to step 2) 2. User indicates “reset” to undo recent edits (branch after step 1)    1. User indicates “reset”    2. Entered information is set back to previous values (return to step 1) |
| Exceptions: | * + - 1. Illegal score is entered (branch after step 1)          1. The score is a negative number or non-numeric          2. The score is higher than the maximum number of points          3. The system indicates that the score is illegal and does not allow score to be saved (return to step 1)          4. The score is set to the maximum number of points (return to step 2)       2. User navigates away from page (branch after step 2)          1. The user navigates away from page after entering one or more scores and before indicating “save”          2. System warns that entered information will be lost and allows the user to stay or go          3. User stays (return to step 2)          4. User goes (use case is terminated)       3. Number of scores do not match the number of students in the offering (after step 2)          1. The system indicates that the number of scores do not match the number of students and no scores are recorded (return to step 1) |
| Includes: | None |
| Priority: | High |
| Frequency of Use: | Number of metrics\*number of students\*number of offerings each semester. |
| Business Rules: | Metric scores cannot exceed the maximum number of points |
| Special Requirements: | None |
| Assumptions: | * For entering multiple grades at once, the order of student names in StOut matches the order of the names in the grade source * Multiple grades will be entered via the clipboard |
| Notes and Issues: | * This use case corresponds to requirement FC7 |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name: | Add/remove students from course offering | | |
| Created By: | AbdulRahman Alduraiweesh | Last Updated By: | AbdulRahman Alduraiweesh |
| Date Created: | 10/9/2017 | Date Last Updated: | 10/9/2017 |

|  |  |
| --- | --- |
| Actors: | Faculty member |
| Description: | Faculty member adding or removing students from offering |
| Trigger: | A student has added or dropped the course and the faculty would like to have the Stout updated accordingly |
| Preconditions: | 1. Faculty member is within a course offering which (s)he is teaching and has created a metric for the offering by entering a description of the metric, its maximum number of points and at least one outcome which it measures 2. There is at least one student in the course offering |
| Post conditions: | 1. The student is added or removed |
| Normal Flow: | 1. Removing a single student.    1. Faculty indicates which student is to be removed    2. Faculty indicates “save”    3. The roster would be updated    4. Feedback to the user indicating success |
| Alternative Flows: | 1. Adding a single student.    1. Faculty looks up student to add.    2. Faculty indicates “save” to commit changes. |
| Exceptions: | * + - 1. User navigates away from page (branch after step 2)          1. The user navigates away from page after adding or removing students and before indicating “save”          2. System warns that entered information will be lost and allows the user to stay or go          3. User stays (return to step 2)          4. User goes (use case is terminated) |
| Includes: | None |
| Priority: | High |
| Frequency of Use: | Depends on the number of students joining or leaving the offer class. |
| Business Rules: |  |
| Special Requirements: | None |
| Assumptions: |  |
| Notes and Issues: | * This use case corresponds to requirement FC4 |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name: | Export a list of students from a course offering | | |
| Created By: | AbdulRahman Alduraiweesh | Last Updated By: | AbdulRahman Alduraiweesh |
| Date Created: | 10/9/2017 | Date Last Updated: | 10/9/2017 |

|  |  |
| --- | --- |
| Actors: | Faculty member |
| Description: | Faculty are able to export a list of students from a specific course offering which they teach |
| Trigger: | Faculty member would like to have a list to be used as a grading sheet. |
| Preconditions: | 1. Faculty member is logged into Stout |
| Post conditions: | 1. A list of students |
| Normal Flow: | 1. Requesting list    1. Faculty navigates to the offering that contains the required list    2. Faculty uses provided link to output names    3. User chooses a directory for comma separated name list to be outputted to |
| Alternative Flows: |  |
| Exceptions: | * + - 1. Empty list          1. If offering selected has no students, feedback would be given to the user indicating that when list is requested.(return to step 1) |
| Includes: | None |
| Priority: | High |
| Frequency of Use: | Depends on how many Faculty use the name list as part of their pipeline. |
| Business Rules: |  |
| Special Requirements: | None |
| Assumptions: |  |
| Notes and Issues: | * This use case corresponds to requirement FC1 |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name: | Add metric to a course offering | | |
| Created By: | ESOF 486 | Last Updated By: | AbdulRahman Alduraiweesh |
| Date Created: | 10/9/2017 | Date Last Updated: | 10/9/2017 |

|  |  |
| --- | --- |
| Actors: | Faculty member |
| Description: | Faculty are able to add a Metric to a desired offering |
| Trigger: | Metrics are added to give a more accurately show performance of students in a particular area |
| Preconditions: | 1. Faculty member is logged into Stout |
| Post conditions: | 1. New metric added to the list of metrics |
| Normal Flow: | 1. Requesting new metric to be added    1. Faculty navigates to the offering a new metric is desired    2. Faculty prompts the Stout to add a new metric and is directed to the correct page.    3. Faculty will add a textual description and fill out the needed fields.    4. Faculty indicates “save” |
| Alternative Flows: |  |
| Exceptions: | * + - 1. Empty Offering          1. If offering selected has no students, feedback would be given to the user indicating that when list is requested.(return to step 3.1)       2. Empty or invalid field          1. If a field is filled out incorrectly or has an invalid character, feedback is sent stating the field and the issue whilst keeping the correct fields intact (return to step 3.1) |
| Includes: | None |
| Priority: | High |
| Frequency of Use: | Depends on how frequent the metrics of an offering needs to be changed. |
| Business Rules: |  |
| Special Requirements: | None |
| Assumptions: |  |
| Notes and Issues: | * This use case corresponds to requirement FC4 |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name: | Edit metric of course offering | | |
| Created By: | AbdulRahman Alduraiweesh | Last Updated By: | AbdulRahman Alduraiweesh |
| Date Created: | 10/9/2017 | Date Last Updated: | 10/9/2017 |

|  |  |
| --- | --- |
| Actors: | Faculty member |
| Description: | Faculty are able to edit a Metric to a desired offering |
| Trigger: | Metrics are able to be edited by Faculty to keep up with any changes or fix an error |
| Preconditions: | 1. Faculty member is logged into Stout |
| Post conditions: | 1. Required metric edited and saved |
| Normal Flow: | 1. Requesting metric to be edited    1. Faculty navigates to the offering where metric needs to be edited.    2. Faculty prompts the Stout to edit a new metric and is directed to the correct page.    3. Faculty will edit the desired fields    4. Faculty indicates “save”. |
| Alternative Flows: |  |
| Exceptions: | * + - 1. Empty Offering          1. If offering selected has no students, feedback would be given to the user indicating that when list is requested.(return to step 3.1)       2. Empty or invalid field          1. If a field is filled out incorrectly or has an invalid character, feedback is sent stating the field and the issue whilst keeping the correct fields intact (return to step 3.1) |
| Includes: | None |
| Priority: | High |
| Frequency of Use: | Depends on how frequent the metrics of an offering needs to be changed. |
| Business Rules: |  |
| Special Requirements: | None |
| Assumptions: |  |
| Notes and Issues: | * This use case corresponds to requirement FC5 |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name: | Delete metric of course offering | | |
| Created By: | AbdulRahman Alduraiweesh | Last Updated By: | AbdulRahman Alduraiweesh |
| Date Created: | 10/9/2017 | Date Last Updated: | 10/9/2017 |

|  |  |
| --- | --- |
| Actors: | Faculty member |
| Description: | Faculty are able to delete a Metric to a desired offering |
| Trigger: | Metrics are able to be deleted if it is no longer used or if it was added by mistake |
| Preconditions: | 1. Faculty member is logged into Stout |
| Post conditions: | 1. Required metric deleted and changes are saved |
| Normal Flow: | 1. Requesting metric to be deleted    1. Faculty navigates to the offering where metric needs to be edited.    2. Faculty prompts the Stout to delete a specific metric and is directed to the correct page.    3. Faculty chooses desired metric    4. Faculty indicates “save”. |
| Alternative Flows: |  |
| Exceptions: | * + - 1. User navigates away from page before saving          1. Feedback is provided to the user warning that his changes have not been saved and will not be saved the page is closed. |
| Includes: | None |
| Priority: | High |
| Frequency of Use: | Depends on how frequent the metrics of an offering needs to be changed. |
| Business Rules: |  |
| Special Requirements: | None |
| Assumptions: |  |
| Notes and Issues: | * This use case corresponds to requirement FC6 |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name: | Viewing student outcome performance | | |
| Created By: | AbdulRahman Alduraiweesh | Last Updated By: | AbdulRahman Alduraiweesh |
| Date Created: | 10/9/2017 | Date Last Updated: | 10/9/2017 |

|  |  |
| --- | --- |
| Actors: | Faculty member |
| Description: | Faculty are able to view the current performance of students in an offering |
| Trigger: | Faculty should be able to view students and how they are progressing in regards to the metrics |
| Preconditions: | 1. Faculty member is logged into Stout 2. Faculty navigates to the offering required for viewing |
| Post conditions: | 1. Page is viewed with the correct requested data |
| Normal Flow: | 1. Requesting scores to be viewed    1. Faculty request a performance view of the offering. |
| Alternative Flows: | 1. Requesting scored to be viewed for different semester than the default    1. Faculty changes the semester before navigating to the offering in which would desired to view |
| Exceptions: |  |
| Includes: | None |
| Priority: | High |
| Frequency of Use: | Depends on the instructors own pipeline |
| Business Rules: |  |
| Special Requirements: | None |
| Assumptions: |  |
| Notes and Issues: | * This use case corresponds to requirement FC9 |

# Explanatory User Interfaces

*[This is an optional section that is used when providing the user with information that could be helpful in understanding the specific requirements in the next section.*

*If this section is used, care must be taken that the general descriptions given here are not presented as requirements.]*

# Specific Requirements

## Functional Requirements

### System (SM)

#### SM1: Login

Given login credentials, Username: Password, the user will be able to sign into their own account. The Login is logged.

Priority: Imperative.

#### SM2: Logout

User indicates they want to logout, system logs and returns the user to the login page.

Priority: Imperative.

#### SM3: Notify

Given a time frame, the system will notify instructors and program coordinator of deadlines on data gathering.

Priority: Important.

#### SM4: Data Lockout

Given a time frame and feature is activated, the system prevents users from adding more data after the time frame as expired.

Priority: Important.

#### SM5: Automatic conversion from operational to analytical data

Given a time frame and feature is activated, the system will prevent new data from being entered, and convert operational data to analytical data.

Priority: High.

#### SM6: Change Log

Upon request, system will show changes made in the past and what they’re connected to; by user, time, what changed.

Priority: Imperative.

#### SM7: User Log

Upon request, system will show user login times, or user privilege(role)/name changes.

Priority: Imperative.

#### SM8: Preserve Data

Given a change in data, the system will not retroactively change old data, instead a change log will be produced.

Priority: Imperative.

#### SM9: Automatic Logout

Given a time frame from inactivity, the system logs the user as timed out and return the user to the login page.

Priority: High.

### Faculty (FC)

#### FC1: Export a list of students from a course offering

The Stout website should be able to output a list of students participating in an offering.

Rationale: Exporting the list of students in an offering may be useful to the faculty member in creating their grading sheet.

#### FC2: Basic Characteristics of Offerings

Once a faculty member is working with a course offering, the course, semester, section and students outcomes of the offering cannot be changed.

Rationale: The course, semester and section are inherent in the offering and changing one of these essentially creates a new offering. The student outcomes cannot be changed because outcomes student outcomes are associated with the course. If student outcomes are to be changed, they must be changed at the course level. The current offering would need to be deleted, the outcomes of the course changed and the offering re-created.

#### FC3: Add/remove students from course offering

Faculty shall be able to add and remove students associated with a course offering which they teach.

Rationale: Students may add the course, drop the course, or not be in the program for which metrics are being collected.

#### FC4: Add metric to a course offering

The faculty member teaching a course offering shall be able to add a metric to the offering. Metric information includes a short textual description of the metric item, the program that the metric will be associated with, the maximum number of points that a student can achieve on the item, and one or more student performance criteria to be measured by the metric. The available performance criteria to be measured will be limited to only criteria that are associated with the program that the metric is to be associated with.

Rationale: Metrics are needed to determine how well students perform on student outcomes.

#### FC5: Edit metric

The faculty member teaching a course offering shall be able to edit the description, maximum number of points, and list of student outcomes which this metric will measure. The student outcomes being measured must be associated with the program that the metric is associated with.

Rationale: Faculty members may reconsider a metric description and the student outcomes it measures, and should be able to modify these. They may have mistakenly entered the wrong number of points for the outcome and should be able to fix their mistake.

#### FC6: Delete metric

The faculty member teaching a course offering shall be able to delete a metric from that course offering. If a metric is deleted, all student scores associated with that metric will be deleted.

Rationale: Metrics may be mistakenly added to the system and the faculty member teaching the offering must be able to clean up the system.

#### FC7: Add student score to metric

The faculty member teaching a course offering shall be able to add a student score to a metric. Student scores must be in the range of 0 to the maximum number of points for the metric. If a score is not entered for a particular student, the score will default to 0.

Rationale: Student scores are needed since they are what enable the system to report the extent to which student outcomes are met.

#### FC8: Edit student score on metric

The faculty member teaching a course offering shall be able to edit a student score on a metric provided the new score is within the range of 0 to the maximum number of points for the metric. If the score is set to anything else (blank for instance) it will default to 0.

Rationale: Student scores may need to be changed due to data entry mistakes.

#### FC9: View extent to which students met student outcome

The faculty member teaching a course offering shall be able to view the extent to which students in the offering met the student outcomes associated with that course. If scores have not yet been entered for any metric(s) which measures a student outcome, the percentage will be 0%.

Rationale: As student scores are entered for metrics, faculty members will want to know the extent to which students met the student outcomes for the default semester or others

### Reports (RE)

#### RE1: Student Weighting

Given indication of student weighting, Output a report that also shows the total number of students

Priority: important.

#### RE2: Section

Given data from each section, Output a report with relevant data within those sections.

Priority: Imperative.

#### RE3: Time Period

Given a time period, Output a report with relevant data within that time period inclusive.

Priority: Imperative.

#### RE4: Programs

Given a program or multiple programs, Output a report with relevant data within those programs.

Priority: Imperative.

#### RE5: Outcomes

Given an outcome or outcomes, Output a report with relevant data within those outcomes.

Priority: Imperative.

#### RE6: Graph options

Given a graphical option selected, Output the report on the corresponding option.

Rationale: A request was made for different graphs for tracking trends and making the information more visual.   
Priority: Imperative.

#### RE7: Data Type

Given a selection of what Report to make, Give the user report options for the report.

Rationale: There are other things that may be tracked in this program other than class outcomes. MAPP being an example.   
Priority: Important.

## Non-Functional Requirements

While the functional requirements detail the functions which the system can perform, the non-functional requirements describe characteristics of the system. These characteristics typically apply to the entire system.

### Design Constraints (DC)

**DC1:** This application is to be developed using MySQL, PHP, HTML, Javascript and Jquery.

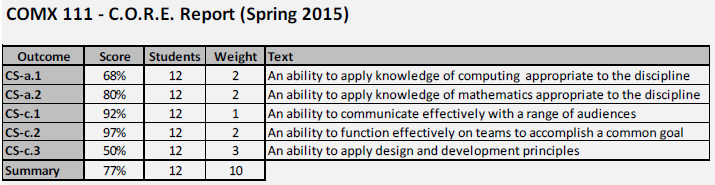
**DC2:** It must be easy to change the value of the metric goal. A programmer should only need to change the code in one place, and all reports will be generated using the new metric goal.

**DC3:** When a faculty member logs into Stout, a list of the course offerings the faculty member is teaching or has taught shall be displayed in reverse chronological order by semester.

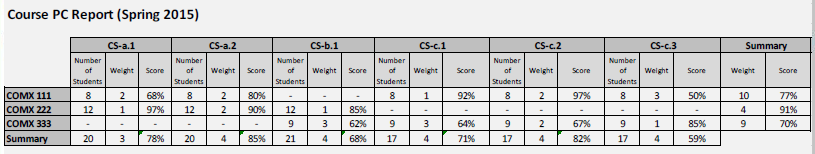
**DC4:** If the faculty member is not assigned to teach any offerings and has never taught any offerings, the system shall state this clearly. Administrative functions shall be accessible from this Faculty View, but with less prominence.

**DC5:** When an administrator logs into Stout the Administrative functions shall be prominent.

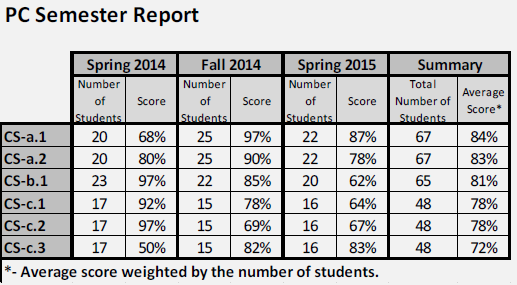
**DC6:** C.O.R.E. Report shall appear similar to the following:



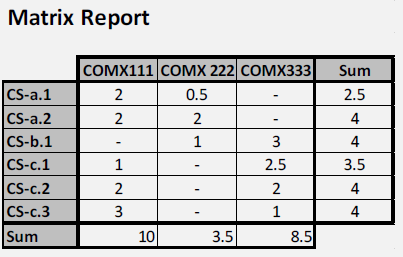
**DC7:** Course PC Report shall appear similar to the following:



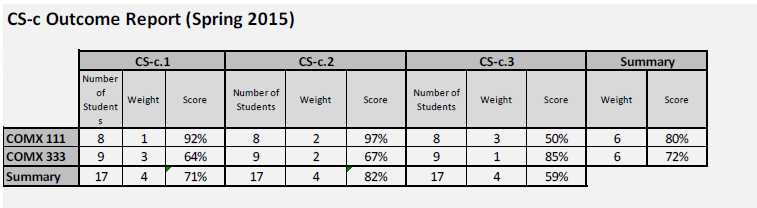
**DC8:** PC Semester Report shall appear similar to the following:



**DC9:** Matrix Report shall appear similar to the following:



**DC10:** Outcome Report shall appear similar to the following:



### Human Factors (HF)

No provisions related to human factors are required by this application.

### External Interface Requirements (XI)

#### Hardware (HW)

**HW1:** The application is accessible on any hardware connected to the Internet which supports one or more of the browsers listed below (SW1).

#### Software (SW)

**SW1:** The system should be able to run on all of the commonly used browsers.(Fire Fox, Chrome, Safari)

#### Communications (CM)

**CM1:** Connection to CAS is required.

### Security (SC)

**SC1:** The application shall only be accessible to MTECHs users within the Stout system.

**SC2:** The application shall not contain Montana Tech student numbers. It may contain student names and will contain assignment, project and/or test question scores.

**SC3:** Stout must comply with FERPA regulations.

### Development Environment (DV)

No provisions related to the development environment are required by this application.

### Standards (ST)

**ST1:** All HTML code adheres to HTML5

**ST2:** All CSS code adheres to CSS 3

**ST3:** A set of coding standards will be used so that the format and character of the code is consistent. These coding standards shall include the W3C standards (<http://www.w3.org/standards/>) for web-content development.

### Delivery Environment (DL)

#### Site (SI)

No site requirements are placed on the application.

### Performance (PR)

### Deliverable Items, Dates and Conditions (DD)

### Cost (CT)

### Quality (QL)

#### Reliability (RL)

*[Reliability is specified as mean-time-to failure of an operational item. An operational profile must be specified.]*

#### Availability (AL)

#### Maintainability (ML)

*[Failures can be classified as occurring in either operational or non- operational delivered items Failures in operational items can be classified by the work products that must be changed to eliminate that failure: code only, code and design, code, possibly design, and requirements. For each class of failure what is the maximum estimated effort required to eliminate that failure and what is the rationale for this estimate.]*

#### Usability (UB)

#### Enhanceability/Extendibility (EN)

*[If the future it might be necessary to change the Functional requirements in specified ways, what is the maximum estimated effort required to make such changes and what is the rationale for this estimate?]*

#### Portability (PT)

*[If in the future it might be necessary to change the above Development or Delivery Environments (DV or DL) to other specified environments, what is the maximum estimated effort required to implement such changes and what is the rationale for this estimate]*

### V&V Activities (VV)

### Database (DB)

*[This optional subsection that specifies requirements for any database to be developed as part of the product. The information in this section will include:*

* *Types of information to be stored*
* *Table attributes (queried, supporting, updated)*
* *Frequency of access*
* *Accessing capabilities and requirements*
* *Data elements and file descriptors*
* *Retention requirements for data.]*

*Care must be taken here to avoid design details. Unless so requested by the client this section should only contain as much information about saved data as is necessary to fully document any of the requirements given above. Since this is the last Non-functional sub-section this sub-section would not appear in the document if the requirements did not involve any data bases information.]*

### Adaptability (AD)

*[If it is specified that in the future it might be necessary to change any of the above Non-Functional requirements, what is the maximum estimated effort required to implement such changes and what is the rationale for this estimate.]*

## Requirements Models

*[This optional subsection, if present, provides models of the functional requirements to aid in clarifying and validating these requirements. A Z language specification is a good example. This sub-section may be skipped entirely if this SRS does not use any requirements models.]*

# Illustrative Use Cases (IUC)

*[This optional section should begin with a hierarchical, logically complete breakdown of all the execution conditions delineated in the functional requirements. Subsections should give detailed use cases for the most important of these conditions. If illustrative use case would not help readers understand the requirements this section should read:* Illustrative User Cases are not developed for this specification.*]*

# Future Enhancements (FE)

*[This section should describe any future enhancements that are contemplated at the time this SRS completed. If there is no known possibility that this product will be enhanced in the future this section should read :* It is not expected that there will be any future enhancements to this product.*]*