**ModuleName**

***Simple Program***

***Development Document
Template***

Montana Tech SE Course

**Developer:**Good Developer

date

Version x.y

Revision Log

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Developer | Comment |
| 1.0 | 7/9/09 | Good Developer | initial version |
|  |  |  |  |

TABLE OF CONTENTS

[1 Introduction 4](#_Toc277518075)

[1.1 Definitions 4](#_Toc277518076)

[1.2 Acronyms and Abbreviations 4](#_Toc277518077)

[1.3 References 4](#_Toc277518078)

[2 Program Perspective, Purpose, and Description 4](#_Toc277518079)

[2.1 Program Perspective 4](#_Toc277518080)

[2.2 Program Purpose 4](#_Toc277518081)

[2.3 Program Description 4](#_Toc277518082)

[3 Program Interfaces 5](#_Toc277518083)

[3.1 User Interfaces 5](#_Toc277518084)

[3.2 Other Intefaces 5](#_Toc277518085)

[4 Program Requirements 5](#_Toc277518086)

[4.1 Functional Requirements 5](#_Toc277518087)

[4.2 Non-functional Requirments 5](#_Toc277518088)

[4.2.1 Implementation Languages (NFR\_LG) 6](#_Toc277518089)

[4.2.2 Design Constraints (NFR\_DC) 6](#_Toc277518090)

[4.2.3 Source Code Standard (NFR\_SC) 6](#_Toc277518091)

[4.2.4 Development Environment (NFR\_DE) 6](#_Toc277518092)

[4.2.5 Operational Conditions (NFR\_OC) 6](#_Toc277518093)

[4.2.6 User Characteristics (NFR\_UC) 6](#_Toc277518094)

[4.2.7 External Intervaces (NFR\_XI) 6](#_Toc277518095)

[4.2.8 Required Work Products (NFR\_WP) 6](#_Toc277518096)

[4.2.9 Delivery Dates and Conditions (NFR\_DD) 6](#_Toc277518097)

[4.2.10 Cost (NFR\_HR) 6](#_Toc277518098)

[4.2.11 Useability (NFR\_UA) 6](#_Toc277518099)

[4.2.12 Reliability (NFR\_RL) 6](#_Toc277518100)

[4.2.13 Availability (NFR\_AV) 7](#_Toc277518101)

[4.2.14 Maintainability (NFR\_MT) 7](#_Toc277518102)

[4.2.15 Adaptability (NFR\_AD) 7](#_Toc277518103)

[4.2.16 Performance (NFR\_UA) 7](#_Toc277518104)

[4.2.17 Quality (NFR\_QU) 7](#_Toc277518105)

[4.2.18 V&V Activities (V&V) 7](#_Toc277518106)

[4.3 Requirements Models 7](#_Toc277518107)

[5 Program Use Cases 7](#_Toc277518108)

[6 Program Test Conditions 7](#_Toc277518109)

[7 Program Test Cases 8](#_Toc277518110)

[8 Program Design 8](#_Toc277518111)

[8.1 Data Data 8](#_Toc277518112)

[8.2 Functional Decomposition 8](#_Toc277518113)

[8.3 Main Program Design Description 8](#_Toc277518114)

[8.3.1 Main Program Design 8](#_Toc277518115)

[8.3.2 Main Program Correctness Argument 8](#_Toc277518116)

[8.4 Function Design Descriptions 9](#_Toc277518117)

[8.4.1 Function Requirements 9](#_Toc277518118)

[8.4.2 Function Design 9](#_Toc277518119)

[8.4.3 Function Correctness Argument 9](#_Toc277518120)

[9 Program Test Data 9](#_Toc277518121)

[10 Program Coding 9](#_Toc277518122)

[11 Program Test Runs 10](#_Toc277518123)

[11.1 Functionality Tests 10](#_Toc277518124)

[11.2 Statement Coverage Tests 10](#_Toc277518125)

[11.3 Performance Tests 10](#_Toc277518126)

[11.4 Reliability Tests 10](#_Toc277518127)

This template provides an annotated outline of the required contents of an *MTM Simple Program Development Document*. All annotation is in blue. No annotation should appear in a specific simple program development document constructed with this template. All sections and subsections that are not annotated as *optional* should be in every simple program development description.

Development of an MTM Simple Program Development Document will proceed in stages. For example the Test Runs section cannot be completed until the module has be coded, debugged, and unit tested.

# Introduction

What is this program for? What larger project is it part of? Are the predecessors that should be referenced?

## Definitions

|  |  |
| --- | --- |
| term | definition |

## Acronyms and Abbreviations

|  |  |
| --- | --- |
| GUI | Graphical User Interface |
| MTM | Montana Tech Method |
| SRS | Software Requirements Specification |

## References

|  |  |
| --- | --- |
| citations |  |

# Program Perspective, Purpose, and Description

## Program Perspective

Is this program a complete application? If not, how does it fit into the larger application/assembly/system?

## Program Purpose

Why is this program being constructed?

## Program Description

Give a brief description of the functionality of this program.

# Program Interfaces

List and logically describe and diagram all interfaces.

## User Interfaces

If prototype user interfaces have been constructed they may be used here as logical models of the user interfaces that will eventually be used in this application. For console applications the user interfaces may be given in the Input/Output functional requirements section

## Other Intefaces

For example, system or other assembly or module interfaces.

# Program Requirements

This section specifies the following requirements for moduleName.

1. Functional Requirements
2. Non-functional Requirements
3. Requirements models

## Functional Requirements

This section should provide a hierarchically list of labeled functional requirements similar to that for a Program Product SRS. The importance of each of these requirements (imperative, important, or desirable – see next section) should be explicitly stated.

A concluding section should provide a table of all the requirements in this section that includesthe section number, the requirement label, a tagging description, and the method to be use for verification: observation, review/inspection, or test.

## Non-functional Requirments

The importance (imperative, important, or desirable) of each of these requirements should be explicitly indicate. Imperative requirements are statements that must be true of the completed product for it to be accepted. Important requirements are expected to be true of the completed product unless the customer/client explictely waives them. Desirable requirements should be true of the completed product unless there is insufficient time or resources to make them so. The customer/client should be informed as early as possible that these requirements will not be met.

Each of these requirements may be further broken down into subsections.

A concluding section should provide a table of all the requirements in this section that includesthe section number, the requirement label, a tagging description, and the method to be use for verification: observation, review/inspection, or test.

### Implementation Languages (NFR\_LG)

Which languages will be used for the source code?

### Design Constraints (NFR\_DC)

Should particular design methodologies and patterns be us ed for this application?

### Source Code Standard (NFR\_SC)

Is there a restriction on which programming languages should be used. Should particular source code standard be used for this application?

### Development Environment (NFR\_DE)

What development process methologies will be used? Is this application to be developed with specific IDEs? How is it going to be unit tested?

### Operational Conditions (NFR\_OC)

On what systems can this program be executed?

### User Characteristics (NFR\_UC)

What characteristics must a user have to successfully operate this application

### External Intervaces (NFR\_XI)

Does this application interface with any external programs or devices other than those normally expected of a Windows program?

### Required Work Products (NFR\_WP)

When this application is completed, what work products should be available and where will they initially stored?

### Delivery Dates and Conditions (NFR\_DD)

When and how are the work products to be delivered?

### Cost (NFR\_HR)

Costs may be stated in dollars or in staff hours.

### Useability (NFR\_UA)

Will any useability reviews or testing be done for this application?

### Reliability (NFR\_RL)

Will reliability testing be used in the development of this application

### Availability (NFR\_AV)

If this application is to run in a specialzed environment are there any recovery and restart requirements?

### Maintainability (NFR\_MT)

This section is included for completeness. In most cases with applications using the MTM no additional explicit, verifiable maintainability requirements are expected.

### Adaptability (NFR\_AD)

Unless possible future enhanacements are listed in the section below there should be no explicit, verifiable adaptability requirements here.

### Performance (NFR\_UA)

It must be possible to verify whatever requirments are given here.

### Quality (NFR\_QU)

This section should state any requirements about system testing and inspections.

### V&V Activities (NFR\_V&V)

This section is included for completeness. In most cases for MTM applications no additional explicit, verifiable maintainability requirements are expected.

## Requirements Models

Any models used to explicate any of the above functional or non-functional requirements should be described here. The requirements explicated by each model should be explicitely listed.

# Program Use Cases

Use cases for consol applications can be included with the functional requirements. Most GUI programs should include use cases that cover the normally expected application behaviors. All behavior illustrated in a use case should be covered by cited functional requirements. For each use case the pertinent functional requirements should be cited.

# Program Test Conditions

Using the requirements given above list all test conditions that the test cases should cover.

# Program Test Cases

List the actually test values that will be used to test each condition. If appropriate annotate the test values as being either a boundary values or an intermediate values. If possible use tables to associate conditions with test values.

# Program Design

Design information should be sufficient for a programmer to code a program that meets requirements. Given only the requirements and design it should be possible to construct a correctness argument.

The subsections listed below should not be taken as mandatory, the subsections used should be appropriate to the type of design and method of design documentation used.

Sections 8.1 through 8.4 that are described here might be used for non-OO designs.

For OO designs this major section should include a class diagram of the design. Subsequent section should provide additional information.

Non-OO or OO design information may be given here or in the code itself. In either case references rather than duplication should be used.

The Design Elements to Requirments Map Correctness Argument subsection should cite at least one requirement for every design element.

## Data Data

List and describe all of the data items that are cited in the Main Program Design Section. Annotated data definition statements from the code may be used.

## Functional Decomposition

List and describe all of the functions used in this design.

## Main Program Design Description

### Main Program Design

Design for main(). Must include pseudocode that satisfies the *MTM Design Language Standard*.

### Main Program Correctness Argument

If required in *Required Work Products* section.

When required, a correctness argument consists of a sequence of C*xx* statements, one for each function/method requirement that provides a programmer-oriented argument for why the design and code for that function/method satisfies that requirement. When required there should always be an correctness argument for main().

## Function Design Descriptions

This section may be omitted if the design for main() does not use any programmer defined functions. If it does there should be separate subsection for each function that is further broken down into Data and Program Design subjsections.

### Function Requirements

Section 4.1 applied just to this function. For very simple functions this section is not required.

### Function Design

Design for *function*(). Must include pseudocode that satisfies the *MTM Design Language Standard*. For simple functions for which the design is obvious from the code this section will not be required.

### Function Correctness Argument

This section will be required only for complex functions or where an argument is need to complete a correctness argument for a higher level function.

# Program Test Data

Provide listing of all the actual test data used in the test procedure, including the "gold" data that show correct test results. Include listings of the test procedures.

# Program Coding

This section is provided for completeness only. Normally this section could simply state:

Program has been coded per the above *Non-functional* specifications.

The program should be coded in accordance the requirements in section 4, the design in section 8. During coding it will often be necessary to modify or further detail the designs in section 8 above so that the design remains isomorphic to the code. The correspondence of code blocks to design elements (when a method/function designs are given) must be explicit and 1-to-1.

# Program Test Runs

## Functionality Tests

Display the procedure and results of functionality tests here.

## Statement Coverage Tests

Display the procedure and results of statement coverage tests here.

## Performance Tests

Display the procedures and results of statement coverage tests here.

## Reliability Tests

Display the procedures and results of reliability tests here.