**A Montana Tech Method Software Development Standard**

**MTM Standard for Small Module Development Overview**

Version 3.1
August 13, 2014

A. Frank Ackerman

*Software Engineering
Montana Tech of the University of Montana*

|  |  |  |  |
| --- | --- | --- | --- |
| *Version* | *Date* | *Author*  | *Comment* |
| 1.0 | 08/22/06 | Frank Ackerman | First version |
| 1.1 | 11/07/07 | Frank Ackerman | Phrasing and added material on various document options |
| 1.2 | 08/19/08 | Frank Ackerman | Correctness arguments and coordination with template |
| 2.0 | 07/02/09 | Frank Ackerman | Refer to project document as *Module Specification* |
| 2.1 | 12/31/09 | Frank Ackerman | Changed name |
| 2.2 | 05/20/10 | Frank Ackerman | Split Dev Doc & Dev Sht processes |
| 3.0 | 05/25/13 | Frank Ackerman | Drop the Simple Program Development Document |
| 3.1 | 08/13/13 | Frank Ackerman | Changed name |

**Montana Tech Software Engineering Students:**

These Montana Tech Method software engineering standards encapsulate Dr. Ackerman’s decades of development experience in the software industry at best-practice organizations; writing, reviewing, and teaching the IEEE software engineering standards world-wide, and many suggestions from various texts and Montana Tech students. These standards have gone through many revisions and additions over several years. They are part of your software engineering studies so that (1) you may have the experience of developing software to a standard (which you may find you need to do if you take a job that requires that you develop high quality software), and so that (2) you will actually have the experience of developing high quality software. You are also invited to participate in the continuing evolution of these standards by studying them critically and making suggestions for their improvement.

Purpose

The purpose of this document is to provide an introduction to the process and applicable standards for developing simple programs according to the Montana Tech Method. See the *Montana Tech Software Development Standards Overview* for a description of simple programs.

Introduction

This *Montana Tech Method for Simple Program Development Overview* describes requirements specification, design descriptions implementation and V&V procedures and standards for developing simple programs. Often such development will be the work of a single developer.

Application

This standard applies to all simple program software engineering development assignments and projects in the Montana Tech Computer Science Department.

Standards

The MTM standards that are applicable to simple programs:

* *MTM Simple Program Development Sheet Template*
* *MTM Design Language for Functions/Methods*
* *MTM Standard for C++ Source Files*
* *MTM Standard for Java Source Files*
* *MTM Standard for C# Source Files*
* *MTM Simple Program Development Sheet Inspection Process*

Process

The entry condition to the MTM development of a simple program is a brief description of the functionality the program is to provide. This may be given in a self-contained problem statement, for example, a problem in a text, or a contest problem, or it may be given in a system/assembly architecture/design document.

The development of a simple program is formally documented in a *Simple Program* *Development Sheet* (SPDS)

## Using a Simple Program Development Sheet

The use of a SPDSshould result in a program with near-zero defects. Furthermore, for small programs that are done in accordance with the SPDS, some of the information may be placed in the code in the applicable following subsections of method/function headers:

DESCRIPTION

REQUIREMENTS

DESIGN

CORRECTNESS ARGUMENT

A simple program development folder should contain test files/scenarios that execute all of the TEST cases defined for that program even if regression test scripts are not written.

## Creating a Simple Program Development Document

1. Make a copy of mtmSimpleProgDevDocTmpltV*x*.*y*.docx and customize it for your program by:
	1. changing the file name,
	2. removing all the explanatory text down to and including the **Simple Program Development Sheet Template** heading
	3. modifying the heading by removing “Template”
2. Each section and subsection of the template contains a description of the material to be supplied for that section. Replace this description with text and diagrams pertinent to your program. You should create test cases before you create a design.
3. Code the program to conform to the design and the MTM coding standard for the specified/chosen implementation language.
4. Compile and debug the program
5. Provide either a requirements to design elements trace, or if required, a correctness argument.
6. Run the specified test cases first to verify functionality, and then again using a test coverage tool if so required. If you are using a test coverage tool examine all unexecuted statements/branches and where appropriate, add to your set of test cases so that all statements/braches that can be invoked with test cases are executed. Where required, also conduct random testing and performance testing.
7. If so required, perform an inspection according to the *MTM Simple Program Development Document Inspection Process* and complete the *Program Inspection Report* document*.*

Assiduously performing the above steps should result in a program with near-zero defects. However, for a variety of reasons few software development projects exactly follow the steps listed above. The following process conditions should, however, be met in most cases:

1. At least start the SPDSbefore starting to code by performing the SPDS Section 3.1 Activities 3.1.1: *MTM document header*, 3.1.2: *Reference to standard version*, 3.1.3: *Brief description of the program*, and 3.1.4: *estimates of program size, effort and delivery date* should be completed.
2. In most cases activities 3.1.5: *Specify the requirements*, 3.1.6: *Construct the logical test conditions*, and 3.1.7 *Select the module test cases* should be done prior to any further development
3. Developers that are not expert in the required or chosen language or development environment may wish to do prototyping coding before choosing and specifying a design. However, activity 3.1.8 *Describe the Design* should be done before coding is complete. As coding proceeds care should be taken to keep the design synchronized with the code. After design, the requirements, test conditions, and test cases should be complete.
4. Activity 3.1.9: *Reference the code* assumes that coding has been complete. The code should compile and correctly execute the requirements use cases.
5. If a correctness argument is not required a requirements to design elements trace should be constructed, otherwise a correctness argument should be constructed (activity 3.1.10).
6. At the end of the day when module completion is formally declared, all of the information that was required should be in the SPDS, the code should conform to the applicable standard, and an inspection, if required, should be completed, including rework, with the final inspection results reported in activity 3.1.14, *Hold an inspection*. (See *MTM Simple Program Development Sheet Inspection Process*)