**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 from “simple program” to “small mdodule” |

**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 small modules according to the Montana Tech Method. See the *Montana Tech Software Development Standards Overview* for a description of small modules.

Introduction

This *Montana Tech Method for Small Modules Development (SMDS) Overview* overviews requirements specification, design descriptions implementation and V&V procedures and standards for developing small modules. Often such development will be the work of a single developer.

Application

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

Standards

The MTM standards that are applicable to simple programs:

* *MTM Small Module 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 Standard for Python Source Files*
* *MTM Small Module Development Sheet Inspection Process*

Process

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

The development of a small module is formally documented in a *Small Module* *Development Sheet* (SMDS)

## Using a Small Module Development Sheet

The use of a SMDSshould result in a module with near-zero defects. Furthermore, for small modules that are done in accordance with the SMDS, 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 small module 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 Small Module Development Sheet (SMDS)

1. From the CS department web site make a copy of the mtmSMDStmplt.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 logical test conditions and accompanying test cases before you create a design
3. You should create at least a top level design before you start to code
4. Code the program to conform to the design and the MTM coding standard for the specified/chosen implementation language
5. Compile and debug the program
6. Provide a requirements to design elements trace, and if required, a correctness argument.
7. 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.
8. If so required, perform an inspection according to the *MTM Small Module Development Document Inspection Process* and complete the *Software Inspection Report* document*.*

Assiduously performing the above steps should result in a module 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 SMDSbefore starting to code by performing the SMDS 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 all the test cases.
5. At the end of the day when module completion is formally declared, all of the information that was required should be in the SMDS or the code and the code should conform to the applicable standard.