BUILDING SIMULATION SOFTWARE: IMPROVING DEVELOPMENT PROCEDURES

Stephanie Mombourquette
Buildings Group
CANMET Energy Technology Centre
Ottawa, Ontario
stephanie.mombourquette@nrcan.gc.ca

ABSTRACT

Without proper standards, Building Simulation Software can be rife with bugs and errors. Increasing software testing, and notably, improving procedures during the software development process itself can help to minimize error and maximize substantially the quality of the software. Therefore, software testing and verification make up a very important part of software validation; however, not to be overlooked are the procedures behind the software development. Version control, documentation (in code as well as external documents) and a good task management system become crucial to the development process, if followed consistently. These practices can improve the software's quality and reliability significantly.

INTRODUCTION

Consider this scenario. You are an engineer managing an incentive program, such as CBIP (Canadian Building Incentive Program). This program provides incentives for builders to build energy efficient buildings. As the manager for this incentive program, you want to have confidence in the simulation software the builders are using. Because you are giving out incentives, the quality of the software and its reliability is very important. What measures can then be taken to ensure the quality and reliability of the software?

Quality assurance of software has many elements. One element of the software development process that can improve the quality is to have good testing practices. Testing software thoroughly will reduce the amount of bugs. Verifying the validity of algorithms used is also vital for good quality software. If algorithms in the software are incorrect, results will be erroneous. Results will then prove unreliable and therefore, useless.

These two quality assurance practices are imperative. Both strive to construct software that is as reliable as possible. There has been a great deal of study done on software quality assurance, with regards algorithm testing and software testing.

Regarding software testing, a paper presented at the eSim 2001 conference entitled "*Developing Project Management and Quality Control Approaches for Improving the EE4 Building Simulation Software,*"[1] discussed the importance of having a good set of testing practices. The paper is focused on a case study that demonstrates how testing improved the quality of the EE4 software. This paper, however, will be looking at a newer and more recent project called Residential Fuel Cell, and will examine the elements behind the scenes, before the testing and algorithm validation, that can, and have, improved the reliability of the software.

John Viega and John McManus from Reliable Software Technologies wrote a paper called "*The Importance Of Software Testing*"[2] and in it they state that "…the more software is tested, the more bugs will be found" and as the bugs are found and fixed, the software is more reliable for the end user. A lot of software on the market exists that will automate testing procedures for your software, such as ApTest[3]. ApTest is customizable software that can easily be modified to offer the best possible testing solution for any given software.

In addition to this, many conferences are held each year related to software testing such as "Software Testing Analysis & Review 2002" in Orlando, Florida[4] and "Software Testing Virtual Conference"[5] which is a continuous virtual conference on-line.

Another element to good building simulation software development is ensuring that the algorithms used are valid. The IEA BESTEST is a series of diagnostic procedures for testing the ability of simulation programs to model the performance of various systems. As such, it tests the validity of algorithms in building simulation software.

The BESTEST procedure is intended to isolate a single facet of a building system, i.e., a particular system algorithm, in each test case, starting with a simplified case and progressively adding complexity. The BESTEST is the Benchmark for **B**uilding **E**nergy **S**imulation **P**rograms[6].

For example, if a developer wanted to test a fuel-fired furnace algorithm, they would run the test case simulations using their software, and compare their results with the results set out in IEA BESTEST from other programs. If the simulated results fall within the range of acceptable results, then the algorithm conforms to test specifications. For an example of an IEA BESTEST test case involving a Fuel Fired Furnace, please refer to Purdy and Haddad (2002). [7]

Much research and work has gone into the study of software testing and validation of algorithms, but a lot of other factors that go on behind the scenes of software development are all too often overlooked. Elements that tend to be overlooked include documentation, managing tasks, software version control and coding standards. Having good underlying software development procedures can be a significant factor in quality assurance and reliability of software, and can ensure that new bugs are not introduced into the software.

This paper will discuss the importance of good development procedures and standards that should be applied during the software development process. The author will focus on why good development procedures are as important as testing and algorithm validation and what the implications are for not using them. Beyond coding standards that establish key guidelines for creating quality software, other development procedures that can maximize the software will be explored throughout this paper. They are:

- Version control
- Documentation
- Task management

WHAT ARE CODING STANDARDS?

Coding Standards are a common set of guidelines for developing and maintaining software. They are intended to be used as a set of consistent coding practices throughout an organization. The main purpose of coding standards is to increase the clarity of the code. Coding Standards vary from organization to organization, as well as from programming language to programming language, depending on what the needs are.

Coding elements that are consistent in coding standards:

- Formatting of the code, such as bracket placement
- Naming conventions:
  - for variables, functions, etc. (i.e. having a maximum length of characters per variable, function etc.)
  - adding a prefix to code, (i.e. "p" for pointers or "k" for constant)
- Good documentation in the code. Making sure that everyone documents what their code does, and a reason for it if it isn't obvious. See Figure 1 for a useful example of how comments can be used to clarify code. If one had to go in and edit one small specific thing, it would take no time at all to look at this code and to know exactly where the change should go.

By adopting good coding standards, the code will be easier to read which in turn will minimize the time and resources that will be required for bug fixing. The code will also be more modifiable and more flexible for quick changes as well as adding new functionality to the software.

In light of this, there is a downside to having great coding practices. Establishing good coding practices involves many resources and is often overlooked due to already existing time constraints. Whether changing existing code to follow the coding standards or writing new code that incorporates them, in the end, taking the time to put good coding practices to use, one can save a lot of time in the long run. When a programmer has to go back to their code months later, or another team member has to edit another's code to add new functionality, it will be easier to navigate through the code and understand how someone coded something, if good coding standards were used. For more information on what coding standards are, refer to "*Developing Project Management and Quality Control Approaches for Improving the EE4 Building Simulation Software.*"[8]

A further example of widely used coding standards is the GNU Coding Standard (stands for **G**NU is **N**ot **U**nix). GNU is an organization that hosts and distributes the code for free software. Anyone can download the source code for software, modify it, add

functionality and resubmit it to the server. Due to the amount of people using this service, it is very important that everyone uses a common set of coding standards. GNU states that their standards are a "…guide to writing portable, robust and reliable programs."[9]

```
/* This program will ask for a name, then return
message in a message box surrounded in stars */

#include <iostream>
#include <string>

int main()
{
  // ask for persons' name
  std::cout << "Please enter your first name: ";

  //reads the name
  std::string name;  //define name
  std::cin >> name;   // read into name

  //build the message that we intend to write
  const std::string greeting = "Hello, " + name + "!";

  //build the second and fourth lines of the output
  const std::string spaces(greeting.size(), ' ');
  const std::string second = "* " + spaces + " *";

  //build the first and fifth lines of output
  const std::string first(second.size(), '*');

  //write all
  std::cout << std::endl;
  std::cout << first << std::endl;
  std::cout << second << std::endl;
  std::cout << "* " <<greeting << " *" << std::endl;
  std::cout << second << std::endl;
  std::cout << first << std::endl;
 return 0;
}
```

Figure 1: An example of some simple code, with comments, and aligned brackets. Note that the comments are in black.

For an organization to adopt good coding standards, it does not require that the standards be written from scratch. Creating an entirely a new document may not be the best approach, time wise, and it would use up a lot of resources that could be focused elsewhere. A method that worked well for the Buildings Group of the CANMET Energy Technology Centre was to obtain permission to use the coding standards of another company that closely fit with their needs. It was then customized for the development team. Using an already existing and applied set of coding standards that have already evolved means the standards are more likely to be complete. This will ultimately result in less time spent testing and establishing the standards, and more time available to complete other tasks.

THREE SOFTWARE DEVELOPMENT STEPS THAT CAN BE IMPROVED UPON

There are 3 steps in the overall software development process, besides testing and algorithm validation, that can be improved upon to ensure quality of any software. Each will be examined separately.

Version Control

Having good version control software such as Visual Source Safe for windows or Teamware for Solaris offers many benefits (See Figure 2). Overall, it improves quality assurance by aiding in reducing problems associated with large projects containing multiple versions and having multiple people working on it simultaneously.

Versioning control software prevents collisions between multiple developers on the same project by preventing multiple edits to the same file at the same time. It permits this by allowing the user to check code in and out of a common repository, or a virtual storage place, which in turn locks files to one person. That person can then compile the code they have changed, and test locally before checking the code back in. This ensures that accurate code is checked in.

One downside to this is that all programmers must check the file out of the repository before any modifications are made to it. If the file is modified before checking it out, this means that the file was never locked while modifications were being made to it. This in turn means that is would be possible someone else edited it in the mean time. The end result would be, when the programmer did finally check in the modifications, it would write over someone else's modifications.

Another way that versioning control software aids in quality assurance is that it also has a labelling feature that allows the user to label a file or a group of files for a product release. By doing this, if for some reason a file or piece of bad code has been corrupted or checked in to the repository accidentally somewhere along the way previous versions of the file can be easily retrieved. The programmer can also roll back completely to another version of the software, or roll back an individual file. One more important feature that most versioning control software has is that it can compare different versions of a file, so that if a roll back for a file is necessary, all changes from one version to the next can easily be viewed and compared. These are features on the programming side of things. The end user has no control, or options to view older

versions of the software. The software, when compiled, does not link to the repository. It compiled only the latest version of the code files.

There are even more benefits to versioning control software. It also tracks who has made changes to each file; in other words, it traces the history for each file as well as the purpose for the changes. When the user checks a file back into the repository, they have the option to insert a comment, for example a version number, or the changes that were made to it. This history can easily be viewed. The importance of labelling files and adding comments will be looked at more closely in "Documentation."

Not necessarily linked to quality assurance but a great benefit nonetheless, and worth noting, is that versioning control software also reduces storage requirements and automates the tracking of files as they change during the course of a development project.

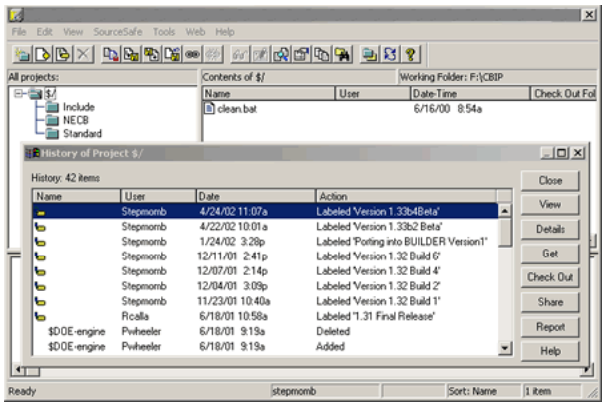

Figure 2: A snapshot of a Versioning control system. This image is showing how a list of a file, or projects history can easily be viewed. When viewing it, one can easily compare two different versions of the same file(s) and even roll back to a previous version.

Documentation

There are two types of documentation that will be discussed, management documentation and in-code documentation. What is meant by management documentation is, external documents that outline certain procedures. One such example could be the procedures for producing a release version of software (See Figure 3). One example of in-code documentation would be the comments one would write in a piece of code to explain what is being done, or how the code is working (See Figure 1).

Documentation inside the code should enhance the clarity of the code that is written. It should be very detailed as to what each section of code does, and what the reasons were for doing it or coding it a particular way. This should enable others to view and understand the code better. See Figure 1 for an example of how comments can be useful in code.

Code walk throughs are also very important. A code walk through is a meeting in which other members of the team get together to see your code and step through it section-by-section and sometimes line-by-line. It is a good process to make sure that the developer wrote detailed and clear documentation as well as clear and properly formatted code, according to the coding procedures put in place by that organization. Sometimes, at the time when you write the code, you may think your notes and documentation are very clear, but 1 month later, you may go back and find, that your notes are not clear, and that you do not understand them.

Code walk throughs are also great for finding bugs, as well as finding potential problems or conflicts within the code. Many heads looking at some code are more likely to spot potential problems or conflicts than one head looking at it. By having a code walk through meeting, others can point out what they don't understand and appropriate notes can be made.

The risk you take by not having a code walk through is that you miss out on having other sets of eyes review your code and point out things you may have missed. By not having others review it, you may introduced new bugs into the software and it will then be more time consuming in the long run to go back and review your code to fix it.

Management documentation is documentation of any procedures that you use during the course of your project. Procedures for coding, writing and compiling help files or even compiling the software itself (See Figure 3). Each team member should be responsible to write the procedures documentation for tasks they are responsible for, and then another team member should revise and test the documentation.

Using the software build procedures for the Residential Fuel Cell project by the Buildings Group as an example, there are three main documents used. These three are not including the other generic documents, such as the coding procedures document, that are used for multiple projects. One Residential Fuel Cell document defines how to write and compile the help files, while another describes how to compile the software, and a third outlines, once the software is compiled, how to create an install file for distribution.

The purpose of any procedural documentation is that anyone of the team members should be able to pick up the document and reproduce the same results every time as well as being able to train new staff easier. By stepping back and defining the software development process formally you can also see potential weaknesses in your procedures, as well as where potential bugs may be introduced into the software.

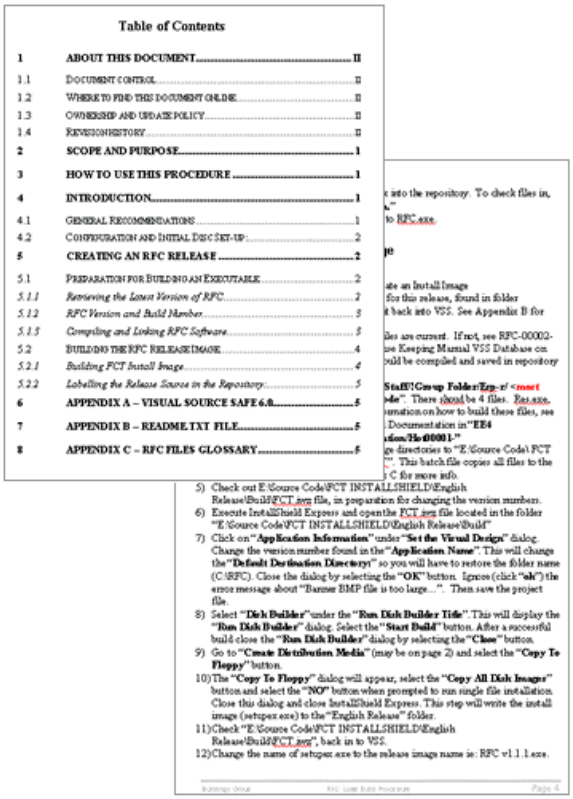

Table of Contents

Figure 3: A snapshot of the Procedures for creating a build (an executable file) of the Residential Fuel Cell Software code. It outlines, from start to finish, how to download the code from the repository, how to make a build of the software, and how to package it into an install executable that the end user will see. These procedures were set into place so that any member of the team can produce the same results every time.

The challenge to having a good documentation system is like that of the coding standards as well as documentation in your code in that it too can also be time consuming. The benefits however do outweigh the cost and risks of not having one at all. In the long run, documenting your procedures saves you or others from trying to remember how to do something every time, or what was done last time, and requires less training time of new employees. Once a good set of procedural documents are in place, your software development should run smoother, and your software, more bug-free, saving you time and resources.

Task Management

The last step in the software development process that can be improved upon is to have a good task management system. A task management system is a system (that can be as simple as a database) put in place for a project to, but is not limited to, report bugs in the software such as who found the bug, who they should be delegated to for fixes as well as the time allocated for each given task. Task management can be a great tool for tracking a project's progress. There are many pieces of software out there that will track your tasks for you. The Buildings Group have opted for something more simple, in-house and less expensive:

- Database with Web interface, for Residential Fuel Cell project. To view info on the Residential Fuel Cell Project, see Figure 4.
- FilemakerPro database with its own customizable interface for EE4 project.

Residential Fuel Cell Project	
Lines of Code	10's of 1,000s
Resources	4 Programmers and 1 Development Support person
Code Walk throughs	3 to date
Total project time from start to finish	1 year

Figure 4: General statistics for the Residential Fuel Cell Project.

Benefits of having a task management system are vast. All task management systems have the same goal, and that is to organize the workload and tasks for a project. See Figure 5 to view a snapshot of the Residential Fuel Cell's "Task Management System."

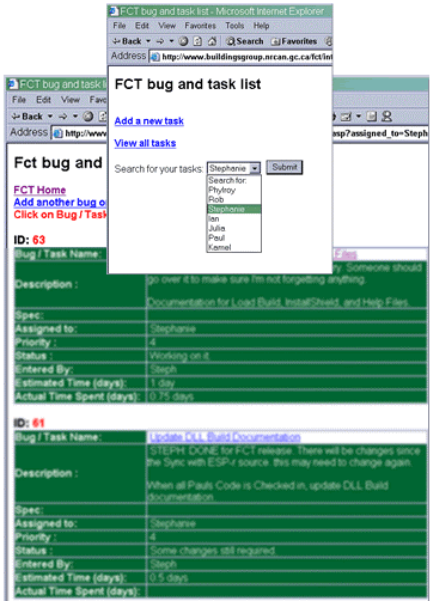

Figure 5: A snapshot of the Residential Fuel Cell Project's "Task Management System". It is a simple Web application (linked to a Microsoft Access Database) in which any member of the team can log in, select their name from a list, and view all tasks that they have been assigned. They can then click on the title of the task, to edit certain fields (either the status, description, or time it took to complete) as they complete tasks.

Using the Residential Fuel Cell project's task management system as an example, any member of the development team can add a new bug or a task to the database through a secure web interface. The project manager will then assign the task/bug to one of the team members as well as a priority, and estimated time it should take to complete the task. Team members can log in to the Web site and search for tasks that have been assigned to them as well as sort by priority. Once the task is completed or if there is something to add, the team member can edit certain fields of the task, such as the actual time it took to complete the task, notes, comments or mark the task as completed.

To create a task management system, you must first look at what it is you want to report. Things that can be important are priority, who the task is assigned to, completion date and for larger projects, perhaps a version (i.e. if you have an English and French version, it will report which one the bug is in).

These are some examples of how a task management system can be used. It helps to organize the project; keep track of the progress of fixes and updates to the code, and ensures that a task gets completed. By having one person delegating tasks, this can prevent multiple people trying to fix the same task, as well as ensures that specific types of tasks get delegated to team members best suited to solving certain problems.

Like most other good software practices, task management can be time consuming. Looking ahead long term, being organized in the area of task management is never a waste of time. Being aware of what your tasks are, your project priorities, and estimating time of completion for each task, allows you to see the "Big Picture." The risks you take by not having a task management system in place are:

- Forgetting to complete a task
- Expend too much time on low priority tasks
- Not spend enough time on high priority tasks.

Another benefit is for future references. If a task comes up that sounds familiar or that was once an issue, you can go back and review the solution.

CONCLUSION

There are many benefits to having good software procedures. The only real downside to putting them into place is that it is time consuming. But in order to have code that is reliable, you need to put time into it, and so much time already goes into bug fixing. Why not make it easier? By putting good procedures into place, in the long run you will actually save time, annoyance and money. For quality improvements to work, resources must be dedicated to them in order to tailor new practices for new challenges.

By not incorporating good practices you risk introducing new bug into your software that can easily be avoided.

Looking at the example given in the introduction, if you are a manager giving an incentive to people, you are going to want to make sure that your software is error-free otherwise people will think they are eligible for an incentive, when they are not and/or you will be giving incentives to people that shouldn't get them or not give it to people that should.

REFERENCES

[1] Rob Calla, Phylroy Lopez, Paul Wyndham-Wheeler. (2001), *Developing Project Management and Quality Control Approaches for Improving the EE4 Building Simulation Software*, eSim 2001 Conference, Ottawa, Ontario, Canada.

[2] John Viega and John McManus, *The Importance Of Software Testing*, 11 January 2000
http://www.cutter.com/consortium/research/2000/crb000111.html

[3] ApTest, *Software Testing Specialists*
http://www.aptest.com/

[4] Software Testing Analysis and Review, *STAREAST 2002: Celebrating Ten Years of Software Success*
http://www.sqe.com/stareast/index.asp

[5] Data Kinetics, *Software Testing Virtual Conference*
http://www.dkl.com/conference/stp/index.html

[6] BESTTEST, *The Benchmark for Building Energy Simulation Programs*
http://software.cstb.fr/iisibat/besttest_us.htm

[7] Purdy, Julia and Haddad, Kamel (2002), O*n Integrating New Models for Building Simulation Software*, eSim 2002 Conference, Ottawa, Ontario, Canada.

[8] See endnote 1.

[9] Richard Stallman et al., *GNU Coding Standard*
http://www.gnu.org/prep/standards_toc.html