NIST Special Publication 1018-5

Fire Dynamics Simulator (Version 5) Technical Reference Guide

Volume 4: Software Configuration Management Plan

Kevin McGrattan
Bryan Klein

In cooperation with:
VTT Technical Research Centre of Finland

National Institute of Standards and Technology
U.S. Department of Commerce

NIST Special Publication 1018-5

Fire Dynamics Simulator (Version 5) Technical Reference Guide

Volume 4: Configuration Management Plan

Kevin McGrattan
Bryan Klein
Fire Research Division
Building and Fire Research Laboratory

April 14, 2009
FDS Version 5.3
SVNRepository Revision : 3782

U.S. Department of Commerce
Gary Locke, Secretary

National Institute of Standards and Technology
Patrick Gallagher, Acting Director

National Institute of Standards and Technology Special Publication 1018-5
Natl. Inst. Stand. Technol. Spec. Publ. 1018-5, 11 pages (March 2009)
CODEN: NSPUE2

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 2007

For sale by the Superintendent of Documents, U.S. Government Printing Office
Internet: bookstore.gpo.gov – Phone: (202) 512-1800 – Fax: (202) 512-2250
Mail: Stop SSOP, Washington, DC 20402-0001

Disclaimer

The US Department of Commerce makes no warranty, expressed or implied, to users of the Fire Dynamics Simulator (FDS), and accepts no responsibility for its use. Users of FDS assume sole responsibility under Federal law for determining the appropriateness of its use in any particular application; for any conclusions drawn from the results of its use; and for any actions taken or not taken as a result of analysis performed using these tools.

Users are warned that FDS is intended for use only by those competent in the fields of fluid dynamics, thermodynamics, heat transfer, combustion, and fire science, and is intended only to supplement the informed judgment of the qualified user. The software package is a computer model that may or may not have predictive capability when applied to a specific set of factual circumstances. Lack of accurate predictions by the model could lead to erroneous conclusions with regard to fire safety. All results should be evaluated by an informed user.

Throughout this document, the mention of computer hardware or commercial software does not constitute endorsement by NIST, nor does it indicate that the products are necessarily those best suited for the intended purpose.

Preface

This is Volume 4 of the FDS Technical Reference Guide. Volume 1 describes the mathematical model and numerical method. Volumes 2 and 3 document past and present model verification and validation work, respectively. Instructions for using FDS are contained in a separate User's Guide [1].

The volumes that make up the FDS Technical Reference Guide are based in part on the ASTM E 1355, *Standard Guide for Evaluating the Predictive Capability of Deterministic Fire Models* [2]. ASTM E 1355 outlines the process of assessing the accuracy of a fire model. Volumes 1 through 3 are the result of this evaluation process. The main purpose of the present volume is to describe the process by which the model software is developed and maintained. A model such as FDS cannot remain static. As progress is made in fire science, the model needs to be updated and improved, but still shown to reliably predict the kinds of phenomena for which it was originally designed.

About the Author

Kevin McGrattan is a mathematician in the Building and Fire Research Laboratory of NIST. He received a bachelors of science degree from the School of Engineering and Applied Science of Columbia University in 1987 and a doctorate at the Courant Institute of New York University in 1991. He joined the NIST staff in 1992 and has since worked on the development of fire models, most notably the Fire Dynamics Simulator.

Bryan Klein is an Information Technology Specialist in the Building and Fire Research Laboratory of NIST. His current focus is on FDS development and user support, along with experimental model validation work.

Contents

Chapter 1

Introduction

1.1 Purpose

The purpose of this document is to describe the policies and procedures for developing and maintaining the Fire Dynamics Simulator (FDS). Such a document is commonly referred to as a *Configuration Management Plan*, known through the rest of this document as "the Plan." This document will be updated as the necessity arises and will establish and provide the basis for a uniform and concise standard of practice for the FDS development process. It is based in part on IEEE Standard 828 [3].

1.1.1 Overview of the Fire Dynamics Simulator (FDS) Software Project

FDS is a Computational Fluid Dynamics (CFD) model of fire-driven fluid flow. The model solves numerically a form of the Navier-Stokes equations appropriate for low-speed, thermally-driven flow with an emphasis on smoke and heat transport from fires. The partial derivatives of the conservation equations of mass, momentum and energy are approximated as finite differences, and the solution is updated in time on a three-dimensional, rectilinear grid. Thermal radiation is computed using a finite volume technique on the same grid as the flow solver. Lagrangian particles are used to simulate smoke movement, sprinkler discharge, and fuel sprays.

1.1.2 Applicability

This document applies only to the program FDS, its companion visualization program called Smokeview, the related NIST publications and Internet site, and the various utility programs that support the two main programs. This document does not apply to third-party software developed by others not affiliated with NIST or its software development collaborators.

1.1.3 Assumptions

The policies and procedures described in this document involve the use of a variety of open source and commercial software and Internet utilities. The continued availability of these resources is an important consideration in the day to day upkeep of FDS and Smokeview. If these resources become unavailable for any reason, the procedures will be updated accordingly.

Chapter 2

Roles and Responsibilities

This chapter describes the roles and responsibilities of the FDS and Smokeview developers.

2.1 Development Team Members

Primary support for the development and maintenance of the Fire Dynamics Simulator (FDS) and Smokeview is provided by the Building and Fire Research Laboratory (BFRL) of the National Institute of Standards and Technology (NIST), an agency of the United States Department of Commerce. The software developers themselves and experimental support personnel work mainly within the Fire Research Division of BFRL. In addition, external collaborators include staff members of the research laboratory VTT of Finland, guest researchers at NIST, and grantees of the NIST extramural fire grants program.

Members of the FDS and Smokeview team share in the development and maintenance responsibilities, which include:

1. Developing new algorithms and improving program functionality
2. Answering user questions
3. Responding to bug reports
4. Issuing periodic updates to the officially released versions of the programs
5. Maintaining the Technical Reference and Users Guides
6. Maintaining a suite of sample cases to demonstrate model use
7. Maintaining a suite of verification and validation cases to test model accuracy and reliability
8. Developing and maintaining a "Road Map" of future development

Most decisions concerning these various tasks are made individually or by consensus of the team members. In the event of a disagreement over technical issues, the decision is made by Kevin McGrattan, leader of the Fire Modeling Group in the Fire Research Division of BFRL, in consultation with team members at VTT, Finland. Non-technical issues are addressed by Anthony Hamins, Chief of the Fire Research Division in BFRL, or by Shyam Sunder, Director of BFRL, depending on the nature of the issue. All policies and procedures described in this document are subject to review by NIST in consultation with VTT, Finland.

Chapter 3

Software Maintenance

Software maintenance consists of the following components: control of the project files and documents, procedures for making changes, and procedures for testing new versions.

3.1 Document Identification and Control

Document identification and control consists of placing all project files in a central location and maintaining a record of changes to those files. The central location is known as the *Repository*.

3.1.1 Project Repository

All project documents are maintained using the online utility Google Code Project Hosting, a free service offered by Google to support software development for open source applications. Google Code uses the Subversion (SVN) revision management system. Under this system a centralized repository containing all project files resides on a Google Code server. Subversion uses a single integer that identifies the version of the entire repository rather than of a specific file (i.e. anytime a change is made to the repository all files are incremented in version number). A record of version number when a specific file was last changed is maintained.

As an open source program, any individual can obtain a copy of the repository or retrieve specific versions repository. Only Team Members described in the previous chapter can commit changes to the repository.

The current location of the FDS repository is `http://fds-smv.googlecode.com/svn/trunk/`. The repository contains the following files:

1. Compiled FDS and Smokeview executables
2. FDS and Smokeview source code files
3. FDS and Smokeview documentation
4. Input files for software testing, verification testing, and validation testing
5. Experimental data files used for validation testing
6. Scripts and post-processing utilities used for software testing
7. Web pages and wikis

3.1.2 Version Identification

At the start of an FDS simulation FDS writes header information to the Smokeview output file, FDS output file, and the FDS log file. This header information contains the version of FDS used to perform that simulation. While each release is tagged with a specific version number (e.g. 5.0.1), there may be many commits of source code, documentation, or other files to the SVN repository before a new version is released with an incremented version number. Thus, if a developer or a user who performs their own compilation between baseline releases discovers an error, the version number written to the output files may not be sufficient to identify the specific set of source code files used. Rather one would need to know the SVN revision number of the most recently committed source file.

This is accomplished by using source file tagging. In each source file a series of character parameter strings are defined. For example, near the top of the file called main.f90 at revision 3144 are lines of the form:

```
CHARACTER(255), PARAMETER :: mainid='$Id: main.f90 3114 2009-01-23 18:15:00Z randy.mcdermott $'
CHARACTER(255), PARAMETER :: mainrev='$Revision: 3114 $'
CHARACTER(255), PARAMETER :: maindate='$Date: 2009-01-23 13:15:00 -0500 (Fri, 23 Jan 2009) $'
```

Figure 3.1: **Example of Revision Identification Strings in main.f90**

The string contains the name of the source file, the version of the file (this reflects only the source file version and not the overall release version number), the date and time of the file version, and the person who checked the file in to the archive. Upon compiling, these strings will be stored in the executable file. The user is then capable of searching the executable file, for example, for strings beginning with `$Id:`. This will result in a list of all source files compiled and their version. Within the SVN archive any specific version of a source file can be extracted and differences between versions can be determined.

Within the source code a series of subroutines were created, one per source file, which parses the `$Id:` in each file and extracts from it the SVN revision number. Each of these subroutines is called at the start of an FDS run and the largest (and hence most recent) revision number is determined. This number is written along with the FDS version number to the output header information. A user can now identify specifically the source code used for a particular compilation of FDS when reporting an error.

3.2 Software Changes

3.2.1 Creating a Change Request

Change requests are submitted using the FDS Issue Tracker. The Issue Tracker is an online service that is part of Google Code. To submit a new issue to the tracker or to provide follow up comments, the user must sign in with a Google Account. The current location of the Issue Tracker can be found through a link under the 'Support' section of the menu on the FDS-SMV website [Link to Tracker].

A change request is initiated by opening a new issue. The issue report contains the baseline identification (version number, compile date, and SVN revision number), operating system, and a description of the defect or enhancement request. Input files or other supporting documentation can be attached.

If the issue is opened by a user, it will be given a status of 'New' until it is reviewed by a developer. If the issue is opened by a developer, the developer can immediate assign a status and an owner.

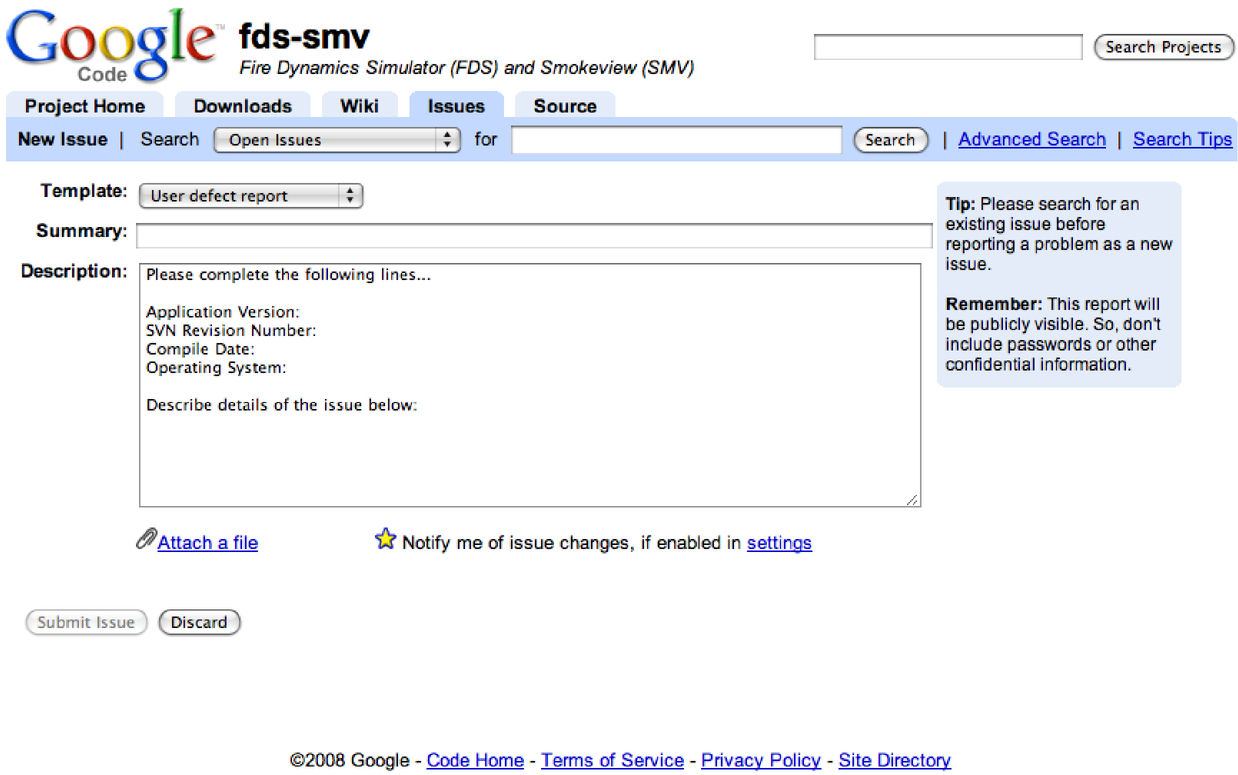

Figure 3.2: **Screenshot of issue tracker reporting form**

3.2.2 Processing a Change Request

A change request may be evaluated by any member of the development team. If the request duplicates an existing request, then the issue status is changed to `Duplicate` and the requestor is sent the issue number of the existing request.

If the request is for an enhancement to FDS, then then the issue type is changed to `Type-Enhancement` and the component type is declared. The request is evaluated for suitability. If it is a valid enhancement request, then the status is changed to `Accepted`. If the request is not to be addressed under the next baseline, then the status will be changed to `OnHold`. If the request is denied, then the status is changed to `WontFix`. An accepted request will be assigned to a developer.

If the request is identifying a defect in FDS, then the issue type is changed to `Type-Defect` and the component type is declared. The defect is evaluated for validity. If insufficient information has been provided in the request, then the status is changed to `MoreInfo` and a description of the additional information required is sent to the requestor. If the defect is due to user error or is the intended function of FDS, then the status is changed to `Invalid`. If the defect is valid, then the request reviewer will change the status to `Accepted` and assign the request to a developer.

Once a change request has be addressed by a developer and changes submitted to the repositoty, the request status is changed to `Fixed`. Once either the requestor or another developer has verified that the changes address the original request, then the status is changed to `Verified`.

3.2.3 Committing Changes

Once a developer has addressed a change request, the modified files are committed to the SVN repository. A description of the changes will be added to the SVN change log. This description first identifies the primary component being changed (for example: FDS Source or FDS Documentation). This component identification will be followed by a brief summary of the changes made. The issue identifier will be included as part of that brief description.

3.2.4 Change Verification

Once a change has been committed and the issue tracker updated to reflect that the issue has been `Fixed`, the changes will be verified. Verification can be done by either the requester of the change or by another developer. Once the changes have been verified to solve the problem reported in the issue, the issue status will be changed to `Verified`. At this point the issue is closed.

3.3 Issuing New Releases

The decision to change versions of the software is made by consensus of the development team, usually after it is determined that enough changes have been made to warrant a new release. There is no formal process to determine when a new release is to be issued. However, once the decision is made, the new version is given a number, it is tested, and then posted to the official download site.

3.3.1 Version Number

New versions of FDS and Smokeview are identified using a specific numbering convention, for example, *FDS 5.2.5*. The version number consists of three integers separated by periods, where the first number is the *major* release, the second is the *minor* release, and the third is the *maintenance* release. Major releases occur every few years, and as the name implies dramatically changed functionality of the model. Minor releases occur every few months, and may cause minor changes in functionality. Release notes can help you decide whether the changes should effect the type of applications that you typically do. Maintenance releases are either just bug fixes or the addition of minor enhancements (such as a new output quantity), and should not affect code functionality.

For each release of FDS there is a unique Subversion (SVN) repository revision number associated with it. For example, FDS version 5.3.0 was built from files in the repository at SVN revision number 3193. The document connecting the release version to the SVN revision number is found in the *FDS Release Notes* wiki page located at `http://code.google.com/p/fds-smv/wiki/FDS_Release_Notes` The SVN revision number allows users to checkout or update their local copy of the project files from the repository to the same state as any previous release of the software.

3.3.2 Testing a New Release

Each proposed release undergoes software testing. There are two sets of input files that are used to test new releases – a verification suite and a validation suite. The verification suite consists of a range of calculations, from ones that just test a feature to ones that compare FDS results with analytical solutions of the governing equations. The validation suite consists of simulations of a wide range of experiments.

Testing will depend upon the type of baseline release: maintenance, minor, or major. A maintenance release occurs every few weeks. Each maintenance release is tested with the verification suite. Because each case requires no more than a few hours of computer time, the suite can be run overnight. The following day, the results of the calculations are automatically plotted and compared with the expected results. This is done simply be recompiling the FDS User's Guide and Verification Guide, each of which contains pictures and graphs that demonstrate the test was successful.

A minor release occurs every three to six months. Each minor release is tested with both the verification and validation suites. The validation cases take considerably longer to run, but the procedure is similar – the results are plotted and compared to the experimental measurements using an automated plotting package that essentially redraws all of the plots and graphs in the FDS Validation Guide. The old and new guides are compared to assess whether or not the new version of the model has become noticeably less accurate than the old. Work is underway to better quantify this procedure.

A major release of FDS occurs every few years. As the name implies, major changes are made to the algorithm, requiring months of beta testing, and several runs through the V&V suites.

3.3.3 Announcing a New Version

Following successful completion of the required baseline testing, a baseline can be released. Prior to release, the version identification information within the FDS source code will be updated to reflect the new baseline. FDS documentation will be updated to reflect the new baseline. The baseline will be compiled and new executable files or installation packages will be placed on the FDS download site. Prior baselines will be deprecated. The current FDS download site is `http://code.google.com/p/fds-smv/downloads/list`.

Bibliography

[1] K.B. McGrattan, B.W. Klein, S. Hostikka, and J.E. Floyd. Fire Dynamics Simulator (Version 5), User's Guide. NIST Special Publication 1019-5, National Institute of Standards and Technology, Gaithersburg, Maryland, October 2007. iii

[2] American Society for Testing and Materials, West Conshohocken, Pennsylvania. *ASTM E 1355-04, Standard Guide for Evaluating the Predictive Capabilities of Deterministic Fire Models*, 2004. iii

[3] Institute of Electrical and Electronics Engineers. *IEEE Std 828-2005, IEEE Standard for Software Configuration Management Plans*, 2005. 1