MFIRE 3.0—NIOSH Brings MFIRE into the 21st Century

A.C. Smith
Office of Mine Safety and Health Research, NIOSH, Pittsburgh, PA, USA

A.F. Glowacki, L. Yuan, L. Zhou, and G.P. Cole
Office of Mine Safety and Health Research, NIOSH, Pittsburgh, PA, USA

ABSTRACT: The MFIRE program, developed in the 1970s by the U.S. Bureau of Mines and Michigan Technological University, has been a mainstay in the modeling of ventilation and fire contaminant spread. It is used by U. S. and international companies to simulate fires for planning and response purposes. However, the original programming language and architecture are considered antiquated by current computer standards. Advances in personal computer operating systems have limited the use of the original program. In addition, the original program is not compatible with modern mine ventilation programs that utilize graphical user interface (GUI) methods. The National Institute for Occupational Safety and Health (NIOSH) completed a major redesign and restructuring of MFIRE under NIOSH contract 200-2009-30794. The program was split into a front-end with a simple GUI, and the MFIRE "engine" back-end. The MFIRE program was rewritten as a discrete event simulation library so it can be used to simulate the progress of mine fires over time. MFIRE's outdated programming language was replaced with an object-oriented C++ approach for ease of future maintenance. A key aspect of the redesign was that third-party developers can obtain ventilation network data from the common memory rather than the default MFIRE data output files. Finally, other improvements to MFIRE were made to increase the size of mine networks that can be modeled and improve capture and processing of runtime errors, adding the ability to report results in both imperial and metric measurement units and to utilize more user-friendly names for data structures.

1 Introduction

The use of simulation software for predicting and assessing, in real time, the impact of fire on the mine ventilation system and the spread of fire contaminants offers significant potential for improved safety and improved chance of success during evacuation and control. The MFIRE program has been a mainstay in the modeling of ventilation and fire contaminant spread. It is used by U.S. and international companies to simulate fires for planning and response purposes.

The MFIRE program had its origin in the MTU-BOM ventilation network code originally developed in the 1970s by Michigan Technological University in cooperation with the U.S. Bureau of Mines.[1] Several upgraded versions were produced, including MFIRE 1.27, 1.29, 1.30, 2.0/2.01, 2.10, and 2.20. The most recent, MFIRE 2.20, was released in 1994. A complete history of the development of the MTU-BOM code and subsequent version of MFIRE is given by Laage et al.[2] MFIRE 2.20 was written in DOS FORTRAN 77; it includes a batch file (MFire.bat) which calls three executable routines (MFire0.exe, MFire1.exe, and MFire2.exe). The executable routines communicate intermediate results via ASCII data files. MFIRE 2.20 consists of approximately 8,600 lines of code. It is a dynamic, transient-state, mine ventilation network simulation program that performs normal planning calculations. It can also be used to analyze ventilation networks under thermal and mechanical influence such as changes in ventilation parameters, external influences such as changes in temperature, and internal influences such as a fire. The program output can be used to analyze the effects of these influences on the ventilation system.

The programming language and architecture of MFIRE 2.20 is now considered antiquated by current computer and programming standards. Because MFIRE 2.20 is a stand-alone program that runs from start to finish without pause, it is difficult to use its ventilation and fire modeling capabilities within larger third-party simulation programs since it is not possible to collect and process intermediate results. In addition, there have been many improvements in fire and contaminant spread modeling since its development. In light of the above, there was a clear need to modernize and improve the program. In 2010, NIOSH recognized this need and completed a major upgrade of MFIRE to bring it into the 21st century. The program was redesigned and restructured into MFIRE 3.0 as follows:

- The outdated programming language was replaced with object-oriented ANSI compliant C++ for ease of future maintenance.
- A back-end was created and designed as a discrete event simulation library to allow for the study of mine fire progression over simulated time, under the control of a front-end program.
- The data architecture was redesigned so third-party developers can utilize ventilation network

data from sources other than the default MFIRE data input files.

- Improvements were made to increase the size of mine networks that can be modeled, to capture and process runtime errors, to report results in both imperial and metric measurement units, and to utilize more user-friendly names for data structures.

This report describes the upgrades and improvements that were made to complete MFIRE 3.0. Ultimately, interfacing improved warning and communication systems and real-time sensor input into MFIRE could provide a comprehensive solution for underground mine fire detection, alarm, and response for the mining community.

2 Technical description of redesign

The MFIRE 3.0 back-end was written in object-oriented ANSI-compliant C++, structured as a class library that can be easily integrated into new simulation programs that need to model mine ventilation and fires. It includes discrete event simulation capabilities that allow mine ventilation and fires to be tracked over time. No improvements were made to the underlying ventilation model. A very simple front-end user interface was developed that uses this class library. This front-end allowed for the testing and validation of the MFIRE library and permits MFIRE to continue to run as a stand-alone program, if desired. However, it is expected that third-party developers will utilize the MFIRE library to develop more sophisticated front-ends (such as adding graphical capabilities to view and edit the mine network and progression of simulated fires). For this reason, the front-end was designed to meet only basic needs for testing and running MFIRE simulations. Further, the class library was designed as public domain software so that third-party developers are not required to purchase additional software or software licenses to utilize MFIRE 3.0.

MFIRE 2.20 ran in a start-to-finish mode without any pauses or delays. Input data, including the events to be modeled, were read from a text file. There were seven types of condition change events allowed:

1. Restore airway (remove fan)
2. Add fan to airway
3. Add fire to airway
4. Remove fire from airway
5. Change simulation time increment
6. Change output time interval
7. Set simulation time for detailed output

These predefined events were processed without pause or delay. One of the seven events allowed users to advance simulated time. In this case, network attributes changed based on the model's calculations, although no event had occurred. MFIRE 3.0 retains this "legacy" mode of operation, including backward compatibility with legacy MFIRE 2.20 input files. In this case, when one event has been processed, the simulation clock is automatically set to the time of the next event in the future events list.

MFIRE 3.0 allows a "new" mode of operation that lets users choose how simulated time is mapped to real time. In this mode, the next event is not processed until a particular clock value has been reached. In addition, this "new" mode adds several new capabilities, as shown in table 1. For example, in between events, it is possible for users to add new events to the queue, poll the back-end for information regarding the state of the system (by providing access to data via common memory), pause and resume the simulation, or end the simulation (emptying the future events list). To support this "new" mode of operation, a new delimited, parsable, input data file in XML format was designed to eliminate the outdated "control card" format of MFIRE 2.20's input file.

Table 1. Comparison of MFIRE 3.0 improvements to MFIRE 2.20 capabilities.

Category	**Description**
1. Programming language and operating system platform	MFIRE 2.20 FORTRAN 77 for the DOS platform. MFIRE 3.0 ANSI-compliant, object-oriented C++.
2. Stand-alone versus class library-based approach	MFIRE 2.20 A stand-alone program that operates from start to finish. MFIRE 3.0 A back-end class library that can be used to embed the MFIRE engine capabilities into future software applications and to facilitate distribution of future MFIRE updates.
3. Support for multi-platform software development environments	MFIRE 2.20 DOS program. MFIRE 3.0 Back-end class library usable by third-party developers who develop front-ends in Microsoft as well as non-Microsoft software development environments.

4. Discrete event simulation	MFIRE 2.20 Runs as a stand-alone program in a start-to-finish mode. All available types of simulation events are created up front in a data input file. New event instances cannot be added without editing this data file and restarting the simulation. MFIRE 3.0 Utilizes a discrete event simulation approach so it can be used to study the progress of ventilation and fires over time. Two modes of operation are supported: • Legacy mode—MFIRE 3.0 reads all data, including events to be modeled, from a legacy MFIRE 2.20 input file. Events are processed sequentially. When one event has been processed, the simulation clock is automatically advanced to the scheduled processing time for the next event in the future events list. • New mode—MFIRE 3.0 reads data, including events to be modeled, from a newly designed delimited and parsable input data file in XML format. It is possible for a front-end program developed using the MFIRE class library to add new events to the queue, poll the back-end for information regarding the state of the system, pause, or end the simulation.
5. Condition change events	MFIRE 2.20 Allows seven types of condition changes. MFIRE 3.0 Implements condition change events as public methods[1]. Because the MFIRE code cannot actually compute accurate results for multiple fires, MFIRE 3.0 prevents more than one fire from being defined.
6. Common memory output access and events	MFIRE 2.20 Not available. MFIRE 3.0 Provides a common memory structure that exposes key back-end data to the front-end to accommodate third-party developers who may need access to back-end data (e.g., to develop graphical displays of the progression of changes to the mine network). It is possible to schedule public events to collect these data at periodic, user-defined intervals.
7. Input files; imperial vs. metric units	MFIRE 2.20 Imperial units. MFIRE 3.0 Legacy mode—Imperial units. New mode—metric or imperial units (but must be consistent for entire input file).
8. Additional input methods	MFIRE 3.0 Public methods for loading all data input to the model (mine network, setup information, etc.).
9. Output files	MFIRE 2.20 A verbose ASCII report containing the results of the simulation for all time intervals. The data in this report are interspersed with comments, making it difficult to utilize/load results into other software packages. All output is contained in one report that is produced at the end of the program run. MFIRE 3.0 A delimited, parsable output file in ASCII format that contains only data (no comments). Verbose ASCII report capability is retained. The user is able to choose either type of output file or both. • Legacy mode—Report(s) are generated at the end of the program run, as occurs in MFIRE 2.20. • New mode—Users can schedule output (of data-only and/or verbose reports) at set time intervals during the program run.

[1] "Public" refers to a section of software code that can be accessed from anywhere in the program (as opposed to "private" code which has more limited access).

10. Naming of variables in source code	MFIRE 2.20 Cryptic. MFIRE 3.0 English-like variable names.
11. Sizing of variables	MFIRE 2.20 Limited because fixed sized arrays are used to store data (junctions, airways, fans, fan curves, etc.). MFIRE 3.0 Maximum size of mine networks is limited only by available memory.
12. Error handling	MFIRE 2.20 Via the output data file. In addition, MFIRE 2.20 replaces user-entered input with its own default values if user input is deemed to be invalid. MFIRE 3.0 Error message reported to front-end when an error is encountered. MFIRE does not replace user-entered data.

3 Front-end

A very simple front-end user interface was developed that uses the MFIRE 3.0 class library. The MFIRE 3.0 front-end is designed as a Windows XP/Vista software application and adheres to standardized Windows user interface design practices. Table 2 describes key features of the MFIRE 3.0 front-end.

Table 2. MFIRE 3.0 front-end capabilities.

Category	**Capability**
1. File input	Provides a means to let a user choose an existing MFIRE data file (in either legacy or new format) to load for use by the simulation.
2. File output	Provides a means to let a user choose file name(s) to use for output. Accommodates the verbose report option, data-only report option, and output from common memory output events. Applicable to both legacy and new mode.
3. Data entry screens	Provides only for new input data needed to support the new mode of operation (legacy input data is still input via the data input files).
4. Error event display	Provides a means to receive and process error messages from the back-end and display them to the user: • Input validation errors—the message informs the user which data field is incorrect. The user is required to correct the bad data before starting the simulation. • Run-time errors—the error message is displayed to the user. Critical errors end the simulation run. Applicable to both legacy and new mode.
5. Common memory output events	The back-end provides capability for the front-end to schedule common memory output events when running in new mode via the data input file or interactively via the user interface. When the front-end receives this data, it writes the requested data to a file(s), pausing until the user elects to continue. User is able to opt to display these results to the screen. Applicable to new mode only.
6. Run simulation	Provides a means for the user to start a new simulation (start the clock and event processing) after valid input has been loaded (applicable to both legacy and new mode). Allows the user to change the way real time is mapped to simulated time in mid-run (i.e., slowing or increasing the speed at which events are being processed). Allows the user to add new events interactively via the front-end. Displays an on-screen clock to track elapsed simulated time. Applicable to new mode only.
7. Pause simulation	Provide a means for the user to pause the current simulation (applicable to new mode only).
8. Resume simulation	Provides a means for the user to continue a paused simulation, whether paused by the user or due to a Common Memory Output display or Error Event display. Applicable to new mode only.

9. End simulation	Provides a means for the user to end the current simulation, which causes all remaining events in the future events list to be deleted and the program run to end. Applicable to new mode only.
10. Save results	Provides a means for the user to stop a simulation and exit the program, saving results so the simulation can be resumed from its present point at a later date.
11. Help menu	A Windows help file is provided that documents usage of MFIRE 3.0 for end users. The help file covers topics needed to use the program, including formatting of input files, other required input events that can be scheduled and how to schedule them via the input file or interactively, and an explanation of on-screen and file output.

4 MFIRE 3.0 Class Library Documentation

An MFIRE 3.0 class library guide is included in the program documentation that can be used by software developers to build new applications. The guide includes usage of all major class constructs, methods, and events. It also includes documenting procedures and options for initializing input and simulation control parameters, creating and scheduling events, simulation time management, and retrieving data results from the back-end.

5 Future Improvements

The modernization and improvements to MFIRE 2.20 that resulted in MFIRE 3.0 were greatly needed. During this process, it became apparent that many additional improvements could be made to the program, both in terms of modeling as well as the program engine. However, these improvements were beyond the scope of the project. Currently, the model is being improved to include parameters such as smoke transport, smoke rollback, and flame spread.[3] In addition, a more comprehensive user interface would greatly enhance the ease of the program's use. However, this too was beyond the scope of the project. Future improvements to the program engine that might be considered are summarized below.

- Convert the unstructured Hardy Cross algorithm and all supporting code to an object-oriented structure more compatible with functional replacements. In particular, eliminate the use of COMMON data and incorporate instead as class elements for each object.
- Add back the water vapor condensation and evaporation algorithms that were part of earlier MFIRE versions, but were removed in version 2.20 and subsequently not converted as part of this project.
- Integrate calculations of fume production rates directly into MFIRE.
- Add some conditional fields to each event, so that specific text or data, conditioned on the output value that triggered the event, can be output through the application programming interface (API). For example, if a ventilation event carries a value greater than X ft^3/min, output an event comment different than if the value is less than X. If a temperature event occurs, perhaps there is a threshold above which the event should indicate that the threshold has been exceeded.
- Currently, the user is allowed to add events to the processing event queue. The ability to edit events in place and see details would allow the user to schedule events and make changes afterwards, before the events have been executed.
- Currently, when creating a fan event, the user can enter five data pairs (air flow and pressure) to define the fan curve. The ability to enter and use up to ten data pairs to define a fan curve would be beneficial.
- Add a network interface to the simulation engine to create a central engine that could be controlled by multiple users simultaneously. This would also provide the ability to remotely monitor several simulations in real time with a simpler user interface (administrative dashboard).
- Incorporate an xml/text editor, or an alternative way to view the configuration file that is loaded/running.
- Currently, documentation of the MFIRE.DLL/API is accomplished with small code snippets and text directions on its use. A visual tutorial of how to use the MFIRE.DLL/API to create a new user interface, by recording the code creation with a verbal description of the process, would be beneficial.
- Add an installer option that would package the needed files together, create user shortcuts, and allow the user the ability to create/change the application default settings. Currently, the MFIRE application and simulation engine are distributed as an executable file, needed libraries (DLLs), and compiled help file.

6 Summary

NIOSH completed a major upgrade of MFIRE to bring it into the 21st century. The program was redesigned and restructured into MFIRE 3.0. To complete this task, MFIRE's outdated programming language was replaced with an object-oriented approach for ease of future maintenance. The back-end was redesigned as a discrete event simulation library so it can be used to study the progress of mine fires over simulated time under the control of a front-end program. The program was

redesigned so third-party developers can utilize ventilation network data from sources other than the default MFIRE data input files. Finally, MFIRE was improved to increase the size of mine networks that can be modeled, to capture and process runtime errors, to report results in both imperial and metric measurement units, and to utilize more user-friendly names for data structures. These improvements have resulted in a more memory-efficient mine ventilation and fire simulation tool that can be more effectively embedded into third-party simulation programs.

7 Disclaimer

The findings and conclusions in this paper are those of the authors and do not necessarily represent the views of the National Institute for Occupational Safety and Health.

8 Acknowledgements

The authors would like to acknowledge and thank Richard Unger, Senior General Engineer, NIOSH, for his suggestions and reviews.

9 References

[1]Gruer RE [1977]. Study of mine fires and mine ventilation, part 1; Computer simulation of ventilation systems under the influence of mine fires. USBM Contract Report No. S0241032. Michigan Technological University, 161 p.

[2]Laage LW, Greuer RE, and Pomroy WH [2005]. U.S. Bureau of Mines training workshop on the "MFIRE" mine fire and ventilation simulator. MFIRE Users Manual, Version 2.20, 110 p.

[3] Zhou L and Smith AC [2011]. Improvement of a mine fire simulation program—incorporation of smoke rollback into MFIRE 3.0. J. Fire Sci. 0(0) 1-11, published online 12 Sept 2011, http://jfs.sagepub.com/content/early/2011/09/01/07349 0411141843 11 p.