An Overview of the MFM Suite for Diagnostic and Prognostic Reasoning of Industrial Process Plants

Harald P-J THUNEM

Systems and Interface Design Department, Institute for energy technology, P.O.Box 173, N-1751 Halden, Norway (harald.p-j.thunem@ife.no)

Abstract: This paper provides the background for and an overview of the development of a software system, the MFM Suite, dedicated to the design and analysis of MFM models related to diagnostic and prognostic analysis of physical processes. The current features of the system are described, as well as some examples of its practical use. The paper also briefly describes how the system facilitates the collaboration between control room and field operators via the Android-based MFM Viewer app.

Keyword: Diagnostics, Prognostics, Multilevel Flow Modelling, Process Analysis

1 Introduction

Multilevel Flow Modeling (MFM)[1] is a methodology for graphical modeling of industrial processes by representing the goals and functions of industrial plants. The purpose is to model the combined functions of any number of physical process components, which together provide the means to achieve one or more goals. The model may then be used for diagnostic and prognostic purposes to determine the possible causes and potential consequences of unwanted process events.

Since MFM models consist of graphical, inter-connected elements arranged in specific structures, there is an apparent need for dedicated software to design them. The MFM models also need to be connected to the process the functionality of which they represent.

Until the development of the MFM Suite, MFM models were designed in MS Visio, which allows a user to drag and drop pre-designed graphical elements to create e.g. flowcharts, organizational charts, fault tree diagrams, and network diagrams. The program further allows a user to define their own graphical elements, in this case MFM functions, and include them in the Visio palette.

Fig. 1 shows Visio with the MFM-specific elements on the left side and an MFM model of a simple windmill in the main window on the right. MFM-elements may be dragged from the left and dropped onto the diagram page, after which they may be resized, rotated and connected to other elements.

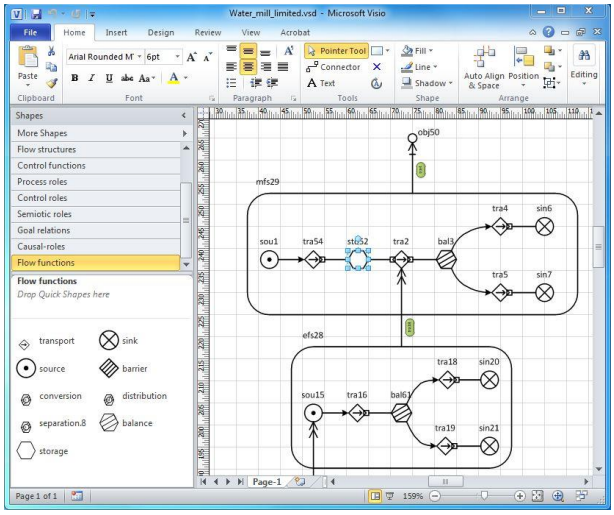

Fig.1 MFM model development in Visio.

Once the MFM model was considered ready to be analyzed, it was exported to a proprietary formatted text file, which was subsequently imported into the MFM Workbench[2]. Developed at the Technical University of Denmark (DTU), this application will perform diagnostic and prognostic analyses on MFM models based on given case settings (i.e. anomalous MFM states). The analysis results are displayed in tree structures (see Fig. 2).

The use of Visio for model design and the Workbench for model analysis results in a cumbersome process, as it is not possible to interactively set the states of the individual MFM functions, either manually or automatically, to call on the Workbench's MFM reasoning functionality, or to display the analysis results directly in the model by e.g. highlighting the

MFM functions diagnosed as the possible causes or consequences of the deviations.

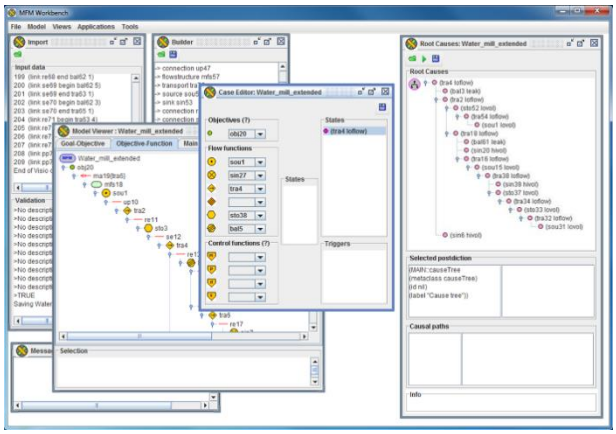

Fig.2 Case settings and analysis result shown in the MFM Workbench.

To perform rapid design and analyses of MFM models, what is needed is:

- MFM-specific graphical design of MFM models
- Seamless connection between designer and reasoning engine
- Interactive case settings
- Analysis results visualized directly in the MFM model.

Furthermore, to perform diagnostic and prognostic analyses on real processes, we require:

- Graphical design of process models
- Real-time acquisition of process sensor data
- Associations between MFM functions and process components/sensors

This paper describes the current status of the MFM Suite, a tool for the design and analysis of MFM and process models[3][4][5]. It is built on the 2nd generation of the Java-based *ShapeShifter* framework[6].

2 Applications in the MFM Suite

The MFM Suite primarily consists of three applications: the *MFM Editor* for graphical creating, editing and verification of MFM and process models, the *MFM Runtime* for real-time acquisition and analysis of on-line sensor data, and the *MFM Playback* for step-wise and simulated real-time playback and analysis of sensor data. Since the applications share a lot of functionality, they are all based on the *MFMApplication* Java class (Fig. 3).

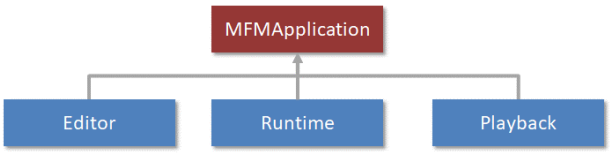

Fig. 3 Inherited functionality.

The MFM Suite also includes a simple launcher from which to start the applications (Fig. 4).

Fig. 4 The launcher application.

3 Editor

The MFM Editor contains two essential graphical modelers, the MFM modeler and the process modeler. As both are based on the ShapeShifter framework, they share a lot of functionality.

3.1 Meta-model display

Most complex processes may be in more than one operating state (e.g. start-up, normal, shutdown), and the functionality of the process components and the process goals may depend on the operating state. Therefore, it may be necessary to create separate MFM models to describe the component functionalities and goals for each of the operating states. However, only one MFM model may be considered *active* at any point, and any MFM reasoning analysis will be performed on the active model only. This furthermore requires that for each MFM model, the alarm limits for process sensors associated with MFM functions must be set individually, as their alarm limits may depend on the operating state.

The MFM Suite applications includes a meta-level display (Fig. 5), which allows MFM models to be connected by arrows, indicating transitions from one operating state to another.

3.2 Click-and-drop model design

Both the MFM and the process modelers provide graphical click-and-drop design of models in a manner similar to e.g. Powerpoint. For both modelers, a list of

available components is shown to the left of the editing area (see Fig. 6 and Fig. 7). Clicking on one of them changes the cursor into a miniaturized version of the component. Clicking anywhere within the editing area will place the component at this point.

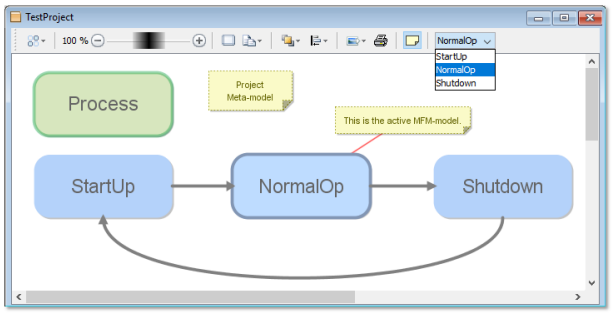

Fig. 5 A project's meta-model.

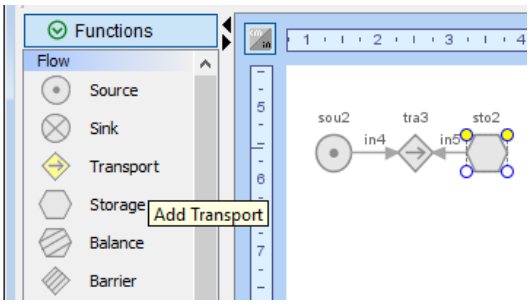

Fig. 6 Design an MFM model by click-and-drop.

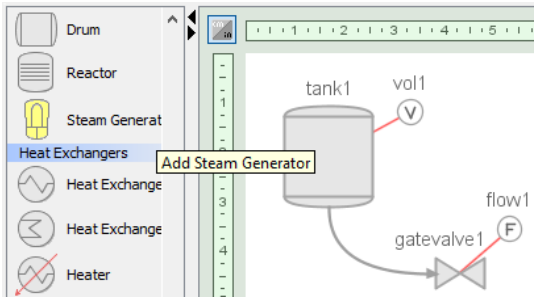

Fig. 7 Design a process model by click-and-drop.

The MFM modeler provides another, faster method which also facilitates the creation of semantically correct models. When selecting an MFM function in the editing area, a list of *allowed* (according to MFM rules) connections (consisting of one MFM relation and one MFM function) will be displayed (see Fig. 8). Clicking on one of them will create the connection from the selected function and place it at a suitable position. The modeler will automatically select the newly added function and update the list of available connections. In this way, complex MFM structures can be designed in relatively few steps.

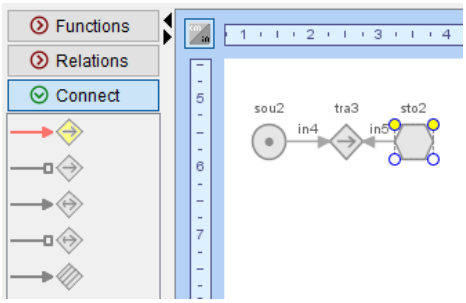

Fig. 8 Assisted design of MFM structures.

3.3 Attaching roles and sensors

Most MFM functions may be associated with one or more roles, which include three *control roles* (*actuator*, *conservant* and *disturbant*), in addition to *agent* and *object* roles. An MFM function can have only one of each role, in addition to a constraint: it cannot have both agent and actuator roles. The association is established by right-clicking on a function and selecting "Attach role" from the popup menu (see Fig. 9).

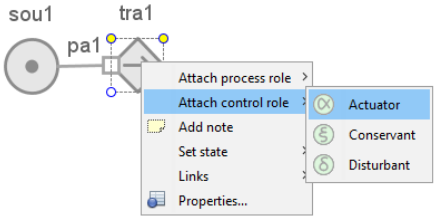

Fig. 9 Attaching roles to functions.

The editor will display a link between a function and its role(s), however this should not be confused with an MFM relation (see Fig. 10).

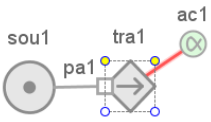

Fig. 10 Displaying roles.

Attaching sensors to process components is similar to attaching roles to MFM functions, by right-clicking on a component and selecting the sensor from a popup menu. Any component may have one of each of four sensor types: *volume*, *pressure*, *flow* and *temperature*. Each sensor is associated with a value and four alarm limits: *high*, *high-high*, *low* and *low-low*, in addition to absolute *maximum* and *minimum* values. The sensors are visualized according to the range of the current value (see Fig. 11).

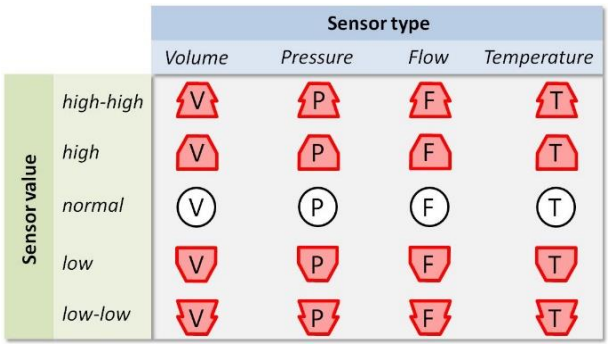

Fig. 11 Sensor display depending on value.

3.4 Connecting MFM functions and sensors

The Editor provides searchable dialog boxes to create association between MFM functions and process sensors, such as e.g. shown in Fig. 12, where storage function *«sto1»* is associated with volume sensor *«vol1»* and transport function *«tra4»* is associated with flow sensor *«flow1»*. A sensor may be associated with different MFM functions in different MFM models depending on which functionality the physical component provides in the given operating mode. The list of associations is therefore stored in separate files with the MFM model for which they apply.

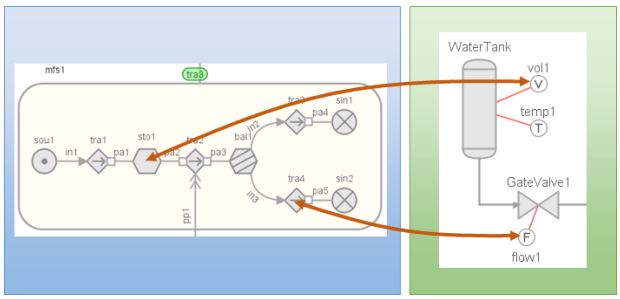

Fig. 12 Associations between MFM functions and sensors.

3.5 MFM patterns

In many cases, the functionality of similar process components may be modeled using identical MFM structures. Designing these structures manually may be time-consuming. The MFM Editor facilitates the reuse of MFM structures through the inclusion of patterns. Any part of an MFM model may be selected and stored in a common *patterns* folder, and the pattern reused using a dedicated dialog box (Fig. 13).

3.6 MFM relation conversion

During the early stages of design of an MFM model, it may be unclear which MFM relations should be used to connect the various MFM functions. Relations can be converted to other types by right-clicking and selecting the desired relation type (Fig. 14). The label will automatically change according to the new type.

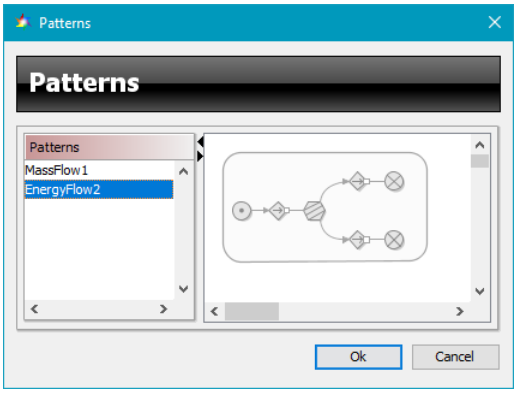

Fig. 13 The pattern dialog box.

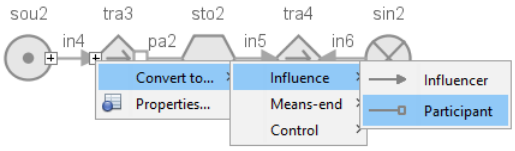

Fig. 14 Converting MFM relations.

3.7 MFM model verification

The editor contains functionality (imported from the MFM Workbench) to verify that an MFM model is designed according to strict rules. When the verification button (✓) in the toolbar is pressed, the editor will invoke the verification method, and any detected errors will be highlighted in the model. As an example, consider the MFM model given in Fig. 15. In this case the direction of participant relation *«pa1»* is wrong. In addition, there are multiple relations connected to the sink function *«sin1»*. A verification of the model gives the shown result, with errors highlighted in the model and both errors and warnings displayed in the *Verification log* (Fig. 16).

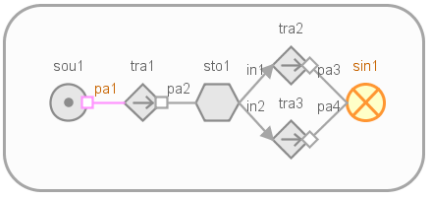

Fig. 15 Visualizing design errors.

3.8 Other common features

Both modelers inherit functionality from the underlying framework, including *copy-paste*, *100-level*

undo/redo, and *shape alignment* and *display order* for improved visualization.

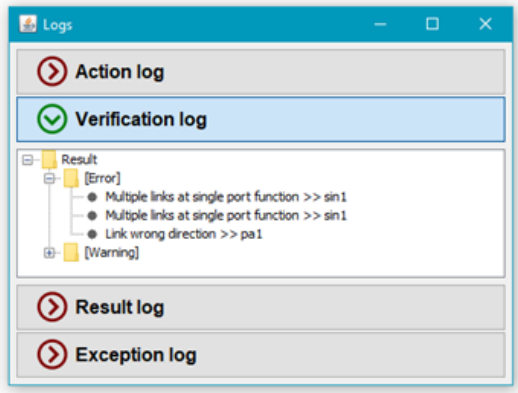

Fig. 16 The verification log.

Furthermore, the framework provides functionality for *color themes*, allowing the user to specify, for each type of graphical element, the fill color, line style/thickness/color and text font/size/color (Fig. 17). The selected theme may depend on the lighting conditions of a room, e.g. in a control room the theme may consist primarily of subdued, «dull» colors so as not to strain the eyes of the operators.

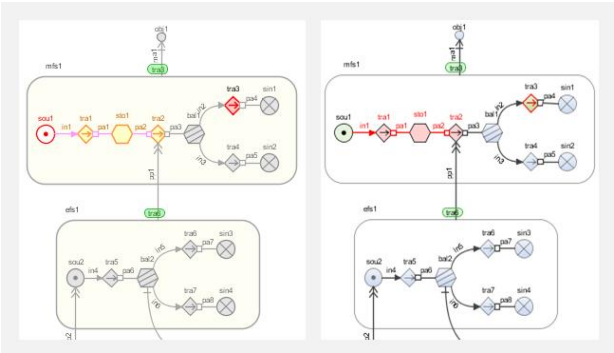

Fig. 17 Analysis results visualized with different color themes.

4 Runtime and Playback

The MFM Runtime and Playback applications use the same ShapeShifter-based graphical components for displaying the MFM and process models as the MFM Editor, however in this case the editing functionality is disabled.

4.1 Analysis functionality

By using the same reasoning engine as the MFM Workbench, the applications will perform diagnostic and prognostic analyses on a given MFM model. The reasoning rules used in both analyses are explained in detail in [7]. The analysis processes will run in separate execution threads, which will set flags to avoid e.g. starting a new diagnosis before the previous has terminated (however, a diagnosis may run concurrently with a prognosis). This will prevent two problems; the analysis process will not lag the sensor sampling in cases where the sampling interval is shorter than the analysis time, and the user interface remains responsive since the main application thread is not blocked (Fig. 18).

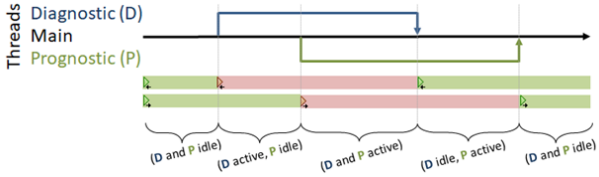

Fig. 18 Concurrent analysis threads.

4.2 Sensor display based on value and origin

In addition to indicating the sensor range by changing the shape, color coding is used to indicate whether the sensor's value is obtained from the process, or whether it is inferred from an MFM model analysis.

As an example, consider performing a cause analysis on the MFM model shown partly in Fig. 19. The sensor *«flow1»* is associated with the transport function *«tra3»*, and the sensor *«vol3»* is associated with the storage function *«sto1»*. Setting the *«flow1»* sensor to a value in the *low-low* range will:

- make the sensor «flow1» be displayed with low-low shape and a red alarm color
- set the «tra3» function to the low-low state and display it with the correct color
- trigger an analysis of the MFM model

Selecting one of the resulting cause paths will highlight all the MFM functions that are part of that cause branch. When selecting the path that contains *«sto1»*, the value of the associated sensor *«vol3»* is set in the correct range and displayed using the correct shape, but this time with the yellow *highlight* color.

If the real cause of the diminished flow through GateValve1 was in fact a low volume in WaterTank, the *«vol3»* sensor should have been shown red. Since it is

not, we can conclude that either the real cause was not a low volume in WaterTank, or there is a malfunction of sensor *«vol3»*.

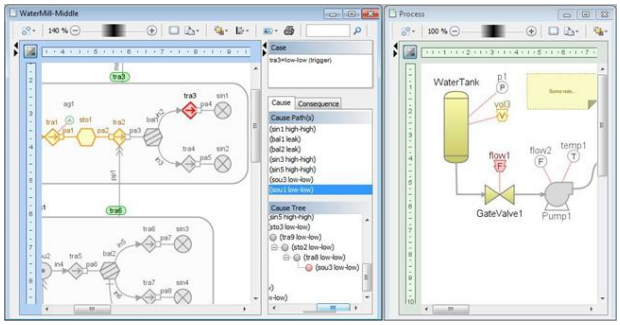

Fig. 19 Sensor visualization.

4.3 Sensor trend display

The process viewer component in the Runtime and Playback applications will display trend graphs for selected process sensors (Fig. 20) based on the Information Rich Display (IRD) philosophy [ref]. Through a dedicated settings menu the user may select which sensor trends to display (searchable selection panel shown), and whether to show different trend colors and markers depending on sensor type.

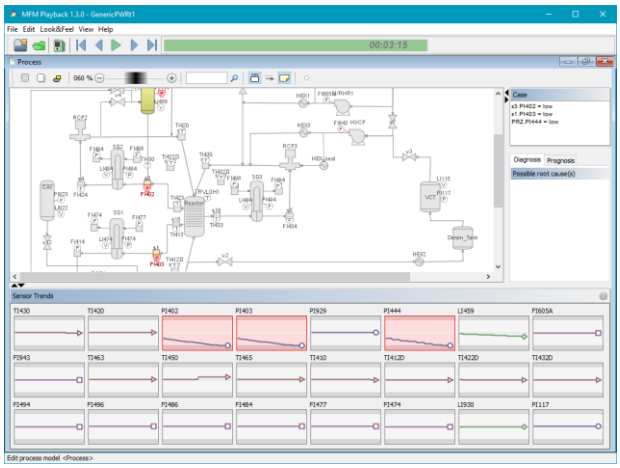

Fig. 20 Process sensor trend panels.

4.4 Data acquisition (Runtime)

The MFM Runtime application can obtain real-time on-line sensor data from an external process or simulator, by using the Java-interface of the *ProcSee* system[8], which is an HRP-developed software tool for developing and displaying dynamic user interfaces.
The parameters required (in particular the names of the ProcSee server and runtime manager) to initiate contact with and retrieve data from a process, are specified from within Runtime, and the parameters are stored in a separate file in Runtime's runtime directory. The process from which to retrieve external sensor data is selected from a drop-down box in the process viewer's toolbar (Fig. 21). The data retrieval is initiated by instructing ProcSee to connect to the selected process and subscribing to the desired sensors. The MFM Runtime is configured to retrieve data from all subscribed external sensors approximately once every second.

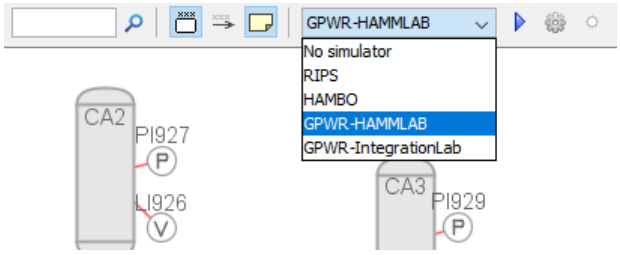

Fig. 21 Selecting an on-line data source.

Whenever sensor data is retrieved, the values of the process model's corresponding *sensor shapes* are updated, as well as any associated MFM functions. If the MFM model containing the MFM functions is in *live analysis mode*, the analysis is performed and any abnormal function states are displayed. The collected sensor data are stored for later retrieval as tab-separated text files.

4.5 Playback of existing data (Playback)

The MFM Playback application will read stored sensor data from files to allow step-wise playback and analysis of previous experiments. The application also includes a slider to move back and forth freely. Furthermore, it can also replay a previous experiment in simulated real-time by activating a timer in a separate thread, and using the stored sampling intervals to update all sensors and re-run any analyses at correct times.

6 Features common to all applications

6.1 Server functionality

The MFM Suite applications include functionality to allow external network clients to download the currently open MFM project (process and MFM models) and to access updated MFM function states

and sensor values. The MFM Suite also accepts commands from external clients for setting MFM function states and sensor value ranges (Fig. 22). This allows e.g. a field operator to have a situation understanding similar to a control room operator and to prune away cause paths that have been determined irrelevant.

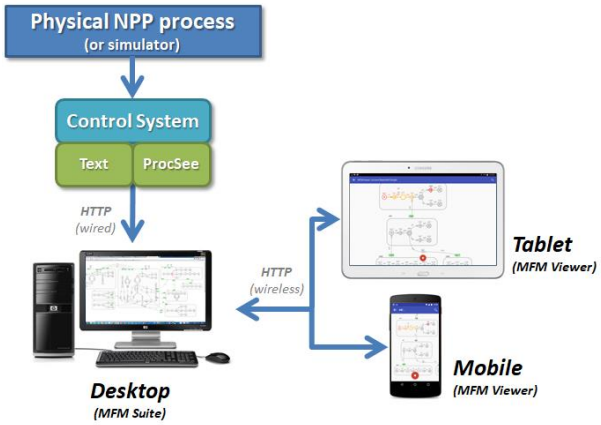

Fig. 22 Data server connectivity.

6.2 Web export

The MFM Suite applications will provide updated displays of models with dynamic data accessible through an ordinary web browser. When opening an MFM project, the applications will generate a number of files, depending on the number of models. An example of the files is shown in Fig. 23.

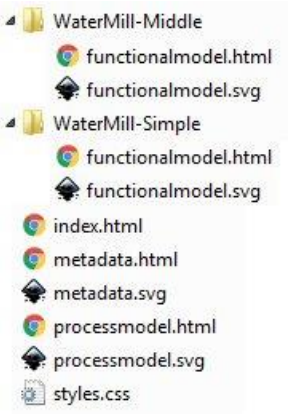

Fig. 23 Web export files.

The graphical models are exported as SVG (Scalable Vector Graphics) files, and since these graphics files are (as the name implies) scalable, they can be zoomed in the browser without the loss of any details (Fig. 24).

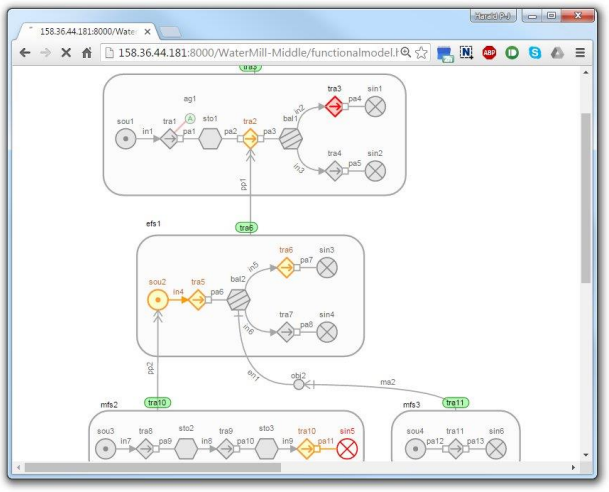

Fig. 24 Web-page showing an MFM model.

Whenever the visual appearance of any element (MFM function, sensor etc.) in a model changes (e.g. due to an analysis affecting the elements), or whenever the user saves the project, the files are updated. In order to keep the display in the browser updated, code is included in the HTML files to reload the pages at given frequencies (default every 5 seconds). This frequency is specified using the Settings dialog box of the applications, which also allows specifying an output folder, to which the generated files are exported. In order to make the web pages available to other devices in a network, this folder could be set to the active root folder of a web server, such as e.g. Apache or IIS.

6.3 Searching

Since both the MFM and process models may contain many graphical shapes representing functions and components, finding individual elements manually could potentially be very time-consuming. The toolbars of both editors have therefore been augmented with textual search fields, which allows the user to quickly highlight elements whose outer label contains the given text string. In Fig. 25 all *storage* functions have been highlighted by entering the search string *«sto»*.

The example further illustrates the importance of establishing a naming scheme for all elements to facilitate the search for specific element types. In MFM models, the built-in naming scheme will suggest that the outer labels of e.g. all storage functions begin with *«sto»*, while for process models the built-in scheme

will suggest that e.g. all temperature sensors begin with *«temp»*.

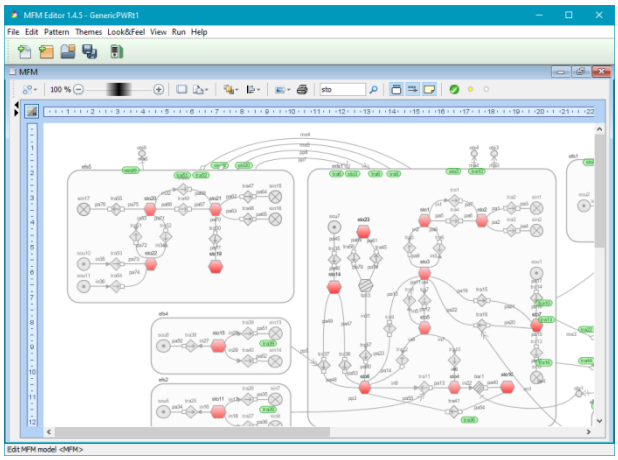

Fig. 25 Visualizing search results.

When creating links (associations) between MFM functions and process sensors, the MFM Editor provides lists of available elements. Since these lists may contain many elements, the link dialog box also includes a search field to limit the number of elements to only those that contain the search text in the outer label.

7 MFM Viewer

The *MFM Viewer* is a graphical model viewer designed for Android-based portable devices such as tablets and cell phones. The use of this application will facilitate a distributed team of e.g. control room and field operators to gain a common understanding of a process situation regardless of their location. For practical purposes a cell phone was used to create the images below, while in a work situation a tablet would be more suitable. The application will scale appropriately to all common display resolutions.

The main menu will allow the user to specify relevant settings, including the IP address and port number of the data server, i.e. any of the MFM Suite applications currently running. The main menu further provides a "Download project" menu item, which will instruct the MFM Suite applications to package the currently open project into a zip file and to transmit the zip file to the device. After receiving the zip file, the MFM Viewer will unzip the project files and display the project meta-model. Clicking on any of the model boxes (MFM or process) opens the model in a new window.

In any of the models, the user may scroll and zoom using common finger gestures.

At the bottom of the MFM and process displays is a green button. Pressing this will cause the application to connect to the MFM Suite and retrieve and display the current MFM states and sensor values at a sampling frequency specified in the "Server Settings". The button will turn red, indicating the retrieval of data (Fig. 26). Pressing the red button or returning to the meta-model will disable the connection. Note that for both models there is a search icon in the upper right corner. Clicking this will open a search field, which allows the user to find (by highlighting) any MFM function or process component whose outer label contains the given text string.

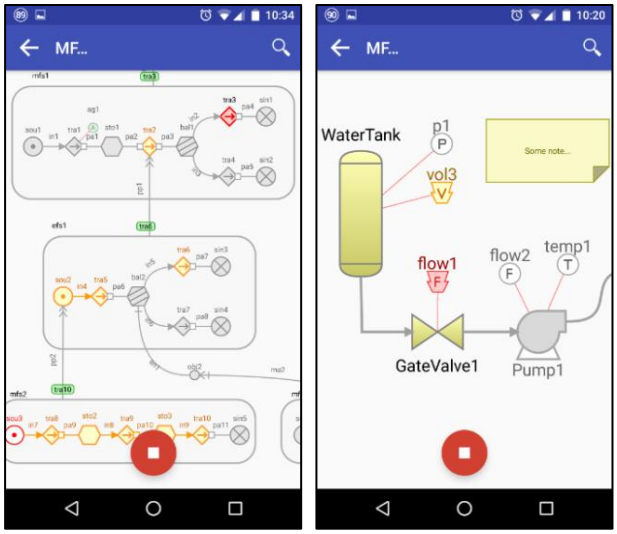

Fig. 26 Displaying models and highlighting analysis results in the MFM Viewer.

The MFM Viewer also allows the user to set individual MFM function states or sensor value ranges. This is done by clicking on the desired element, which will open a dialog box, and clicking on the desired state or value Fig. 27). This will in turn trigger a re-analysis of the models in the MFM Suite application.

8 Examples of use

Several tests have been conducted using an MFM model of the primary side of two PWR simulators; the *RIPS* simulator which is based on the Ringhals power plant, and a *generic* PWR simulator[9][10][11]. Several scenarios were tested, including reactor trip cause by

too low pressure in the steam generators, and a high pressure in the reactor coolant system.

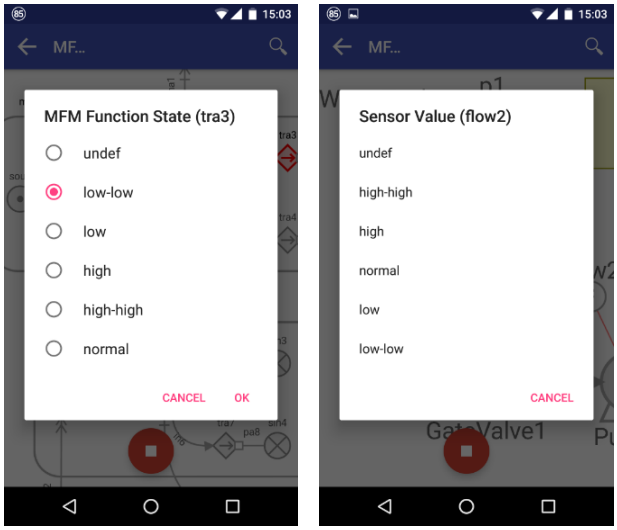

Fig. 27 Setting MFM function states and sensor ranges in the MFM Viewer.

Before running the experiment, the simulator was initialized to a normal running state. At this point, the values of all sensors were registered and used to set the individual alarm limits. The process model contains a total of 49 sensors, 21 of which are associated with MFM functions.

Before running each scenario, the simulator was reset to a normal running state. The control room operator would run the selected scenario by adjusting selected process parameters and letting the simulator run its course.

In most cases, the reasoning engine performed flawlessly. Fig. 28 shows one scenario with multiple anomalous sensor readings (right side) and the successful diagnosis using MFM reasoning (left side).

It was, however, noticed that in some cases the selection of trigger function in the MFM model would cause a memory problem (stack overflow) in the reasoning engine. By manually selecting another trigger function, the problem would disappear.

The MFM reasoning engine uses the Java-based JESS inference engine[12], which employs recursion, a common cause of stack overflow problems. It is therefore reasonable to assume that the selection of different trigger functions will result in different recursion levels during the rule-based reasoning. One solution is to increase the Java call stack; however, the maximum size of the call stack may be platform dependent and difficult to ascertain.

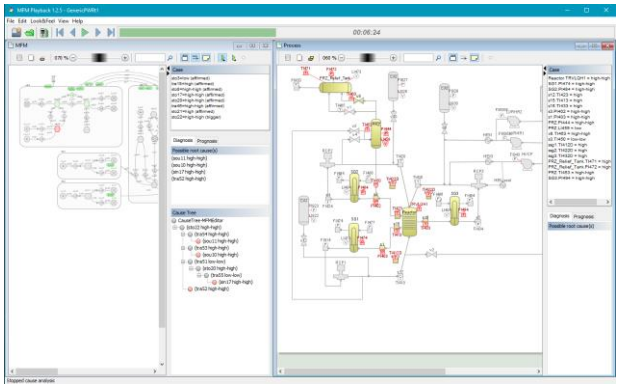

Fig. 28 Scenario diagnostic.

Since the JESS system is no longer supported, the MFM reasoning engine has been re-implemented using the Drools inference engine, however, the experiments have not yet been re-run to verify any performance improvements.

9 Further work

The primary focus on the MFM Suite development has until recently been on including necessary functionality to graphically design MFM and process models, create links between them, and perform online analysis with simulated process sensor data. When presenting the analysis results, focus has primarily been on how the MFM functions are affected, and which MFM functions have been identified as root causes Interpreting the results has required some familiarity with the MFM methodology, which is unreasonable to expect from e.g. a control room operator.

To make the analysis results more accessible to someone without MFM training, the focus of the analysis presentation will therefore shift towards the process display, indicating which process components are the root causes and which components are affected by the identified cause paths.

The activity will further evaluate various display modes to enhance the operators' comprehension of a given process situation in light of established analysis results, and to facilitate collaboration between control room and field operators via dedicated interaction mechanisms.

References

[1] M. Lind, “An Introduction to Multilevel Flow Modeling”, International Journal of Nuclear Safety and Simulation, 2(1), pp. 22-32, 2011.

[2] M. Lind and X. Zhang, “Agent Based Reasoning in Multilevel Flow Modeling”, Proceedings of First International Symposium on Socially and Technically Symbiotic Systems, 2012.

[3] H. P-J Thunem: "Use-Case of an Agent-Oriented Framework for Supervision, Diagnosis and Prognosis Applications", Proceedings of the combined ESREL2012/PSAM11 conference (pp. 4910-4917, ISBN 978-1-62276-436-5), June 25-29, 2012, Helsinki, Finland.

[4] H. P-J Thunem: “The development of the MFM Editor and its applicability for supervision, diagnosis and prognosis”, Proceedings of the ESREL2013 conference (pp. 1807-1814, ISBN 978-1-138-00123-7), Sept 29 - Oct 2, 2013, Amsterdam, Holland.

[5] H. P-J Thunem and X. Zhang, “The continued development of the MFM Suite and its practical application on a PWR system”, Proceedings of the ESREL2015 conference (pp. 2463-2471, ISBN 978-1-138-02879-1), Sept 7 - 10, 2015, Zürich, Switzerland.

[6] H. P-J Thunem, A. P-J Thunem and M. Lind, “Using an Agent-oriented Framework for Supervision, Diagnosis and Prognosis Applications in Advanced Automation Environments”, Proceedings of the European Safety and Reliability (ESREL 2011) conference (pp. 2368-2375, ISBN 978-0-415-68379-1), September 18-22, 2011, Troyes, France.

[7] X. Zhang, “Assessing Operational Situations”, PhD thesis, Department of Electrical Engineering, Technical University of Denmark, June 2015.

[8] H.O. Randem, H. Jokstad, T. Linden, H.Ø. Kvilesjø, S. Rekvin and A. Hornæs, “ProcSee - The Picasso Successor”, Halden Work Report HWR-799, OECD Halden Reactor Project, September 2005.

[9] X. Zhang, H. P-J Thunem, M. Lind, S. B. Jørgensen and N. Jensen, "Practical Application of the MFM Suite on a PWR System: Modelling and Reasoning on Causes and Consequences of Process Anomalies", Halden Report HWR-1118, Institute for Energy Technology, August 2014.

[10] X. Zhang, M. Lind and H. P-J Thunem, "Applying MFM to the PWR Analysis: The experimental results and preliminary conclusions", Halden Report HWR-1191, Institute for Energy Technology, March 2016.

[11] H. P-J Thunem, "Diagnostic Decision Support – Recent Development and Updates of the MFM Suite ", Halden Report HWR-1223, Institute for Energy Technology, August 2017.

[12] E. Friedmann-Hill, “Jess in Action”, Manning Publishing Co (ISBN 1930110898), USA, 2003 (http://www.manning.com/friedman-hill/)