INL/EXT-20-60549

Light Water Reactor Sustainability Program

Report for 2.2.1 Task 5: Develop and Document a State-Based Alarm System for a Nuclear Power Plant Control Room Using Machine Learning

Jens-Patrick Langstrand, Robert McDonald, Hoa Nguyen

November 2020

U.S. Department of Energy

Office of Nuclear Energy

INL/EXT-20-60549
Revision 0

Light Water Reactor Sustainability Program

Report for 2.2.1 Task 5: Develop and Document a State-Based Alarm System for a Nuclear Power Plant Control Room Using Machine Learning

Jens-Patrick Langstrand, Robert McDonald Jens-Patrick Langstrand,

November 2020

Idaho National Laboratory
Idaho Falls, Idaho 83415

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

IFE/INL-196543

SOW 14512

Report for 2.2.1 Task 5.a Develop and document a state-based alarm system for a nuclear power plant control room using machine learning

Institute for Energy Technology

Address	KJELLER NO-2027 Kjeller, Norway	HALDEN NO-1751 Halden, Norway
Telephone	+47 63 80 60 00	+47 69 21 22 00
Telefax	+47 63 81 63 56	+47 69 21 22 01

Report number IFE/INL-196543-2.2.1	**Date** 2020-11-05
Report title and subtitle Report 2.2.1 for Task 5.a: Develop and document a state-based alarm system for a nuclear power plant control room using machine learning	**Number of pages** 31
Project/Contract no. and name Contract no 196543	
Client/Sponsor Organisation and reference Idaho National Laboratory, USA	
Abstract	

	Name	Date	Signature
Author(s)	Jens-Patrick Langstrand Robert McDonald Hoa Nguyen	2020-11-05	
Approved by	Stine Strand	2020-11-05	

HR-e-rapport-e ver 2007-04-24.1

1. Background

Institutt for Energiteknikk (IFE) operates the Organization for Economic Co-operation and Development OECD Halden Reactor Project. The organization has extensive experience from more than 20 years of research in human system interface (HSI) design and the operation of nuclear power plant research simulators in the Halden Man-Machine Laboratory (HAMMLAB).

HAMMLAB serves two main purposes. These are the study of human behavior in interaction with complex process systems and the development, testing, and evaluation of prototype control centers and their individual systems. The aim of HAMMLAB is to expand the knowledge of human performance in complex process environments in order to adapt new technology to the needs of the human operator. By studying operator performance in HAMMLAB and integrating the knowledge gained into new designs, operational safety, reliability, efficiency, and productivity can be improved.

IFE also provides new and innovative technology to customers in the form of operational task-based displays, large screen displays, innovative eye-tracking programs, and innovative performance testing methods. IFE also provides expert support in both nuclear power plant operations and setting up and running operations-based experiments and workshops.

2. Introduction

Idaho National Laboratory (INL) has contracted IFE to support human factors research and the development of leading-edge technology to support control room operators as the U.S. fleet undergoes the modernization and digitalization of legacy plants and control rooms. As the nuclear industry starts to shift to more digital controls and systems, these upgrades provide more information to the main control room in the form of digital signals and values. Also, vendors provided an increase in the number of alarm points. New digital control rooms shift from main control boards to soft controls and from alarm panels to a single display. This change to a single display has created an alarm waterfall problem, a situation in which the addition of new alarm points causes an overload of information to the operator during either a plant disturbance or other than normal full power operations. IFE’s goal is to find a workable solution to assist operators in both handling and understanding incoming alarms during emergency situations under these waterfall conditions.

3. Current Design

In legacy control rooms, operators might have 20 to 30 alarms to deal with on multiple control boards throughout the control room during a reactor trip. The new digital alarm system in the upgraded control room will display between 150 and 200 alarm points on a single screen. This overload of information prevents the operators from quickly identifying abnormal alarms, without having to scroll through the alarm list looking for abnormalities. One way to tackle this problem would be to create a state-based alarm system that would recognize plant states, identify alarms that are expected in those states, and suppress expected alarms from the alarm screen so that unexpected alarms would be displayed, prioritized, and easily identified by the operators. This can be done today, but the process is labor intensive in that an operator or training instructor would be required to run scenarios and log those alarms that need to be suppressed. The hope is that machine learning (ML) can provide an alternative, less labor-intensive solution.

4. Machine Learning for Unexpected Alarm Detection

The creation of traditional state-based alarm systems requires operators or trainers to manually specify the state conditions for the reactor at each state. This process differs in that, instead of trying to manually define these states, data is collected and ML is used to learn the state conditions automatically. The research described in this report is a continuation of the research performed last year where supervised learning and semi-supervised anomaly detection was used to establish that ML could, as a proof-of-concept, be used to develop state-based alarm solutions (Langstrand, Nguyen, and McDonald, 2019). Based on the results of the testing performed last year where anomaly detection on simple scenarios performed well enough to demonstrate the feasibility of this ML approach, this year's research was focused on this technique.

Anomaly detection is useful when accessible data contains predominantly positive or negative samples. In this case, the aim is to model the nominal behavior of the reactor and detect any unusual behavior in new data. This approach can also be called semi-supervised learning because only one class or type of data is used during the training of the model.

5. Scope

The core deliverable is a report on the use of ML to create a state-based alarm system as compared to a human subject-matter expert (SME) state-based alarm system. Due to a COVID-19 outbreak during the early part of 2020 and issues with access to the simulator used by the initial SME, we were forced to adapt the initial plan to a modified process. The project used the generic pressurized-water reactor (gPWR) simulator found in both INL's Human System Simulation Laboratory (HSSL) and IFE's HAMMLAB in Halden, Norway. We will be able to compare IFE research results to the gPWR simulator results in HSSL. The results contained in this report are easily repeated in the HSSL using the same initial conditions and running through the same procedures.

The whole process, from collecting data to training and deploying a ML system, will be covered in this report, as will lessons learned.

6. Data Collection

In order to train ML models, we first need to collect meaningful data that capture the nominal behavior of the gPWR. We chose a number of simple scenarios, including reactor startup from zero power, low power plant shutdown with manual turbine and reactor trips, and plant cooldown and depressurization. Once the nominal behavior of the gPWR was validated, we used more complex scenarios to further test the feasibility of the ML approach. Those scenarios included: manual reactor trip and manual safety injections, and reactor at full power with reactor trip, including safety injection due to malfunctions. Choosing these scenarios allowed us to focus on testing and iterating our ML models on a simpler problem, rather than initially spending an inordinate time on scenarios and complex combinations. Initially, we believed that we would need to create a data collection tool for this project; however, it turned out that we could repurpose an IFE-developed tool.

6.1. Data Collection Tool

To collect data from the gPWR, the IFE's data collection tool was used. It is a tool developed by IFE that allows the collection of synchronized sound, video, simulator signals, and simulator events as well as having the ability to replay collected data. Only data relevant for the modeling of nominal reactor behavior were collected such as: process signals, alarm signals, and process events. In total, we ran the simulator 20 times with scenarios of various lengths and complexities.

6.2. Scenario Descriptions

6.2.1 Scenario 1

The initial condition (IC) on the gPWR is IC 112 "Ready to remove RHR during heat-up." The controlling procedure is GP-002, Section 5.0, Step 55. This plant state is a normal condition during start-up activities after a refueling outage. The initial reactor coolant system (RCS) temperature is 342ºF and 312 psig. The crew has secured the B Train of the residual heat removal (RHR) system and is ready to secure the RHR A Train and continue heating up the plant to greater than 350ºF. There were 55 initial alarms at the start of this IC, as shown in Figure 6. This scenario was identified as a bad candidate for this ML project due to the time between different component operations. Also, there were few changes to the plant and no significant changes to the alarm display. This scenario is still relevant with the use of ML to identify the expected alarms, but, for this project, we were looking for more changes in the alarms, both clearing and activating, so we could identify if the ML was able to identify unexpected alarms.

6.2.2 Scenario 2

The IC on the gPWR is IC 16 "Reactor start-up Shut-down Banks out." The controlling procedure is GP-004, Section 5.0, Step 30. An estimated critical condition of 95 steps on Control Bank D has been calculated. The crew is ready to start withdrawing the rod in accordance with the procedure until the reactor has reached criticality. This state-based condition is common after every reactor trip or outage. This is the normal startup procedure and process. There are 30 initial alarms at the start of this IC, as shown in Figure 7. This scenario was also identified as a poor candidate for this ML project due to the time that is required to start up the reactor and the lack of changes to the alarm display. ML could still be used to identify both expected and unexpected alarms. We chose not to continue with this based on time constraints.

6.2.3 Scenario 3

The IC on the gPWR is IC 5 "Ready to trip the turbine during Shutdown." The controlling procedure is GP-006, Section 5.2, Step 34. This scenario is a normal shutdown from 8% reactor power followed by a turbine trip and then, when reactor power is less than 3%, a manual reactor trip is initiated. The crew will then continue a cooldown to 520ºF and depressurization to 1,900 psig. There are 11 initial alarms, see Figure 8, at the start of this IC. The crew has reduced the reactor power to 8% and are ready to trip the turbine. Per the procedure, when the crew is ready, they trip the turbine manually and continue with the procedure. After the turbine has been tripped, there will be 23 alarms in the alarm summary, shown in Figure 9. Once reactor power has decreased to less than 3%, the crew is directed by the procedure to trip the reactor. After the reactor trip, there are 29 alarms in the alarm summary, as shown in Figure 10. The crew then transitions to GP-007 "Normal Plant Cooldown." This procedure is used to cooldown the RCS to 520ºF and reduce the RCS pressure to 1,900 psig. The crew will follow this procedure until they have reached the previously desired plant conditions and then the scenario is stopped. The final alarm list contains 40 alarms, as shown in Figure 11. This scenario is easily replicated with near identical results, and is a good scenario to work with the ML. The scenario lasted approximately between 40 to 60 minutes, depending on the rate of cooldown, and the number of alarms increased from 11 to 40.

6.2.4 Scenario 4

The IC on the gPWR is IC 1 “Middle of Life, 100% reactor power Xenon is at equilibrium.” The controlling procedure is GP-005, Step 143. This scenario is an abnormal situation with an inadvertent reactor trip and inadvertent safety injection. There are five initial alarms at the start of this IC, as shown in Figure 12. An inadvertent trip will be initiated, followed by an inadvertent safety injection. This scenario was created to see if ML could handle the large volume of alarms created by the reactor trip and safety injection. There are 105 alarms displayed on the alarm screen, as seen in Figures 13.1–13.3. By using an inadvertent trip and safety injection, we establish a base event where there was no cause for the trip or safety injection. This will provide a good base condition so that, when there are significant reasons, the unexpected alarms will stand out and give the control room staff a better understanding for the cause of the trip or safety injection.

6.3. Final Scenarios with Faults

6.3.1 Scenario 3

Scenario 3 was conducted using the ML program connected to the gPWR simulator. The scenario was conducted as described in Section 6.2.3. Figures 14, 15, 16, and 17 show the results of the four stages: initial, after turbine trip, after reactor trip, and after cooldown. Figure 18 shows the results after faults FT-494C (XMT), and FT-494C Steam Generator C Component Steam Flow, Channel III final severity 0.00, and CFW16A Main Feedwater Pump #1 Trip is inserted. ML identified the expected alarms in the initial condition and in the turbine trip phase of the scenario. When the reactor was tripped, the ML identified all alarms as expected except one, the “Reactor Trip Manual.” This alarm was later identified as an expected alarm, as the crew proceeded with the cooldown and depressurization of the RCS. While the crew continued with the cooldown and depressurization, nine additional alarms were received, which ML identified as three expected alarms and six unexpected alarms; however, all received alarms were expected. When the faults were inserted into Scenario 3, there were three additional alarms associated with the faults. The ML correctly identified that these alarms were not expected. A fault was inserted to see if the ML could handle a sudden influx of alarms and not change the state/condition selected, in this case a normal reactor shutdown and cooldown. We inserted fault MSS01B Steam Line Break Inside Containment (SG#2). Figures 19.1 and 19.2 show the results of this induced fault. The results were that the ML maintained the state/condition and identified almost all new alarms as unexpected while maintaining the previously identified alarms from earlier in the scenario. There were 41 initial alarms prior to the insertion of the final fault, then, after the fault, there were 89 alarms. One of the new alarms was identified as expected because it was identified earlier in the scenario and continued to be expected in the selected state/condition. The ML identified 46 additional alarms as unexpected for the state/condition.

6.3.2 Scenario 4

Scenario 4 was conducted using the ML program connected to the gPWR simulator. The initial condition is IC 1, and the condition/state is a 100% power inadvertent reactor trip with a safety injection. The fault installed is the RCS18B RCS18 Small CL LOCA B Loop 100% = 4.5-inch-diameter break. The final severity was set to 25% with a ramp time of 5 minutes and a delay of 2 minutes. The severity and ramp time were selected based on the desire for the leak to be large enough to cause a reactor trip and safety injection but small enough to allow time for the plant to respond to changes. The ramp time allows the pressurizer and charging system a chance to respond to changes in the RCS/pressurizer's pressure and level. These are not expected alarms in the initial scenario, so the goal is to see if the ML can identify these initial changes as unexpected. The next step is to see if ML can then separate a normal reactor trip and safety injection and identify the alarms that were unexpected. The goal here is to have those alarms that are unexpected to standout so the operator can easily determine the cause of the reactor trip and safety injection. Figures 20.1–20.3 show the results of the ML response to the fault. The scenario starts with a radiation monitor alarm that is not expected in this state/condition. Before the reactor trip at 05:55, there are some expected alarms and unexpected alarms. The unexpected alarms would help the crew focus on the pressurizer. Once the reactor trip has occurred at 06:10, there is a safety injection due to low pressurizer pressure. This is not expected in the selected state/condition. Of the first 50 alarms that are received, seven are unexpected in the selected state/condition. These 7 unexpected alarms show the cause of the both the reactor trip and the safety injection. In alarms 51–77, there are three unexpected alarms, which show a problem inside the containment building as the relative humidity is rising. Alarms 78–105 show an additional 26 alarms that are unexpected and continue to point to a problem with the RCS in the containment building. There are 36 unexpected alarms in this scenario. To a crew, reducing the number of meaningful or unexpected alarms from 105 to 36 would ease the burden of diagnosing the event and ensure that they take the proper actions in the emergency procedures.

6.4. Data Description

The data collected by the IFE data collection tool are received in multiple files, so only the active directory monitor files used will be covered here:

- valueInfo.txt: describes the content of the binary file containing all the simulator process signals. This binary file assigns every process signal a unique identifier, lists the valid ranges for that signal, its engineering units, and also includes a short description.
- values.dat: contains all the process signals collected during a simulator run in binary format.
- processEvents.txt: contains all the process events with timestamps and identifications (IDs), for instance lamps turning on, valves opening or closing, and alarms on, and an operator can easily identify the cause for the reactor trip and safety injection. The unexpected alarms can provide a quick insight into the cause of the abnormal condition during the simulator run.

6.5. Deliverable 1: Data Preprocessing Script

Before using the collected data to train the ML models, the data was preprocessed into a format that could be used by the ML models. The binary files containing the collected process signals were parsed into a csv and further packed into pickles, which allows for a quicker loading of the data. See Figure 1 for an image with graphs of some of the process signals collected during this step.

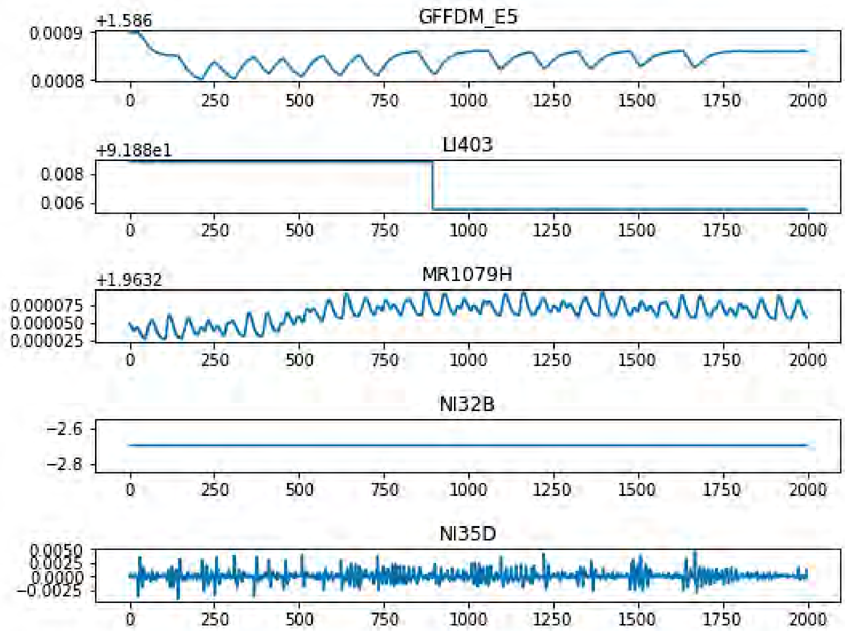

Figure 1. A selection of process signals used to train the ML systems.

The alarm signals were extracted from the process events file, using the timestamps, signal IDs, and alarm states to convert the events into continuous signals. The timestamps of both data sources were used to synchronize process signals with alarm signals.

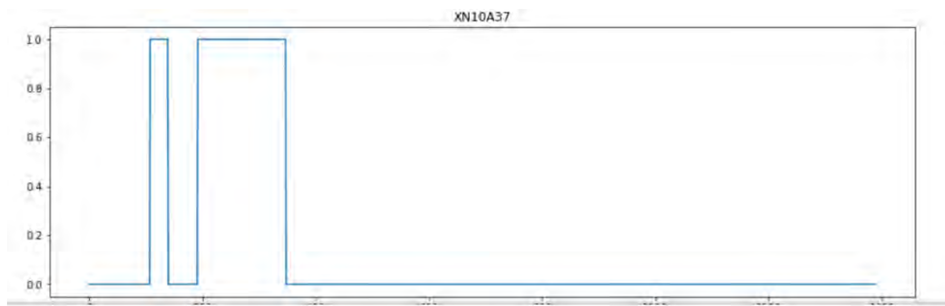

Figure 2. An example of an alarm signal that was created using the process events where a value of 0 represents the alarm off and a value of 1 represents the alarm on.

Both data sources were loaded as data frames into a Python notebook script. In total, there are 655 process signals and 751 alarm signals. Finally, the signals required normalization in order to make the ML training more effective and reduce the amount of time required to train a model. Without this step, the training would require more time and might even fail to converge on a good solution. An attempt to use the specified ranges from the file describing the ranges of each signal to normalize the values failed; however, it failed because some of the signals had incorrect ranges specified. As a result, min-max scaling was performed on the available data in order to have data within a valid, normalized range. Ideally, the predetermined ranges would have been used to normalize the input data because the ranges cover all possible values that the signals can have. In this way, even if the full range is not represented in the training data, the model would have access to the full range when working on unseen data.

Because the written script relied on data collected by the IFE-developed data collection tool, it would be difficult for INL to directly utilize the script without having Viewer access.

However, one option could be to extend the real-time tool to also have data collection capabilities, which would allow INL to collect data.

6.6. Machine Learning Modeling

With the data preprocessed and normalized, it was possible to begin ML modeling using the collected data. The data was split at the scenario level so that a model could be trained and tested for each scenario. The resulting datasets were highly imbalanced in terms of having 755 potential possible alarm signals, but only a fraction of these signals were ever in an on state. To mitigate the class imbalance problem, the root-mean-square error was used as the loss function during training. It is more effective than the mean-square-error in penalizing prediction errors in unbalanced datasets. Otherwise, the model might predict that all alarms are in an off state and still achieve a high accuracy. The dataset was then split into batches with a sequence length of 120 samples. The order of the batches was randomized before splitting the batches into training, validation and testing sets. As the training process ran, the batches were sampled at a random starting point to collect sequences of the desired time frame, which in this case was set to 30 samples. When pulling data from the simulator, it has a frequency of about three samples per second. Meaning that the model was trained to take an input sequence of 10 seconds and predict the expected alarm states from that.

Once models trained on the individual scenarios performed well, a combined scenario model was trained, utilizing data from both scenarios. Real-time testing was performed to determine how well the models filtered expected alarms and highlighted anomalous alarms. During testing, the operating procedures defining the scenario were followed and showed that the models could filter many expected signals. Additionally, anomalies were introduced during the scenarios, and the models were able to highlight these without filtering any by accident.

6.7. Tools and Frameworks

To handle the data preprocessing and preparation as well as ML, Python was used with frameworks, such as Pandas, Numpy, and Keras. Keras is a widely used ML framework that provides a higher-level, easier-to-use API for modeling. Keras also supports multiple popular ML libraries, such as Theano, CNTK, and Tensorflow. Tensorflow was selected for these tests. Tensorflow is a popular open-source ML library, developed by Google.

6.8. Modeling

Long short-term memory (LSTM) layers were used during modeling, as they have been shown to be effective in processing sequential data. This layer type also performed better than fully connected or convolutional layers, which were tested during last year’s project. An autoencoder-type architecture was used where the initial layer is large, then shrunk in size (encoded), and then expanded back to the original size (decoded) in the last layer. This means that the large number of signals coming into the network will be encoded into a smaller dimension and then recreated from the encoding in the later layers. See Figure 3 below for an overview of the final model architecture used.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input (InputLayer)           [(None, None, 655)]       0
_________________________________________________________________
lstm (LSTM)                  (None, None, 1024)        6881280
_________________________________________________________________
lstm_1 (LSTM)                (None, None, 512)         3147776
_________________________________________________________________
lstm_2 (LSTM)                (None, None, 512)         2099200
_________________________________________________________________
lstm_3 (LSTM)                (None, 1024)              6295552
_________________________________________________________________
output (Dense)               (None, 751)               769775
=================================================================
Total params: 19,193,583
Trainable params: 19,193,583
Non-trainable params: 0
_________________________________________________________________
```

Figure 3. The architecture of the final LSTM model used during the evaluation of the testing scenarios.

Once trained, the model is used as follows. The 655 process signals are entered into the network, and the network predicts the expected state of the 751 alarm signals. The resulting output of the model is then subtracted from the current alarm state. A difference close to zero means that the signal is expected. If it is bigger than some defined threshold, it is treated as unexpected. This threshold was set to 0.8 during the real-time testing.

6.9. Results

There were a series of test runs to see how the ML would respond to the two identified scenarios. The results are seen in Figures 21–27. These test runs were conducted using the gPWR in HAMMLAB.

Once the desired state/condition has been identified, a scenario can be created around that state. The scenario creation process usually takes 4 hours to set up and test the scenario for the first time and determine if it will be stable and work for the desired time. Once the scenario has been defined, it will need to be run and the data captured using IFE's data collection tool. The scenarios vary in length, as Scenario 3 was 90 minutes whereas Scenario 4 lasted between 10 and 15 minutes. During testing, we shifted back and forth between scenarios to verify that the ML could quickly identify the desired state and expected initial alarms. The ML never had any problems identifying the initial set of alarms, and the results show the success in the final scenario runs.

6.10. Deliverable 2: Trained Machine Learning Models

As part of the tool described in the next section, some of the trained ML models have been included for testing purposes. There are some steps required to go from a newly trained model to one that can be deployed and used in an application:

- First, the model must be exported, either to two files, with one describing the model architecture and one containing the model weight, or to one file containing both the architecture and the weights.
- Second, the model must be converted from a Keras model (.h5) to a frozen Tensorflow model (.pb).

- During this step, it is important to find the name of the input and output layers of the model, as these will be required when deploying the model for inferencing later.

6.11. Deliverable 3: Simulator and Machine Learning Interface Tool

To connect trained models to the simulator so that the model can receive live data and perform alarm filtering, a tool in C# was developed that uses ProcSee to communicate with the simulator (see Figure 4). ProcSee uses a publication and subscription model that allows it to subscribe to the process signals of interest and receive messages containing the value of these signals approximately three times per second. Using the publication module, the output of the ML model is published to update an alarm display made for the testing the alarm system.

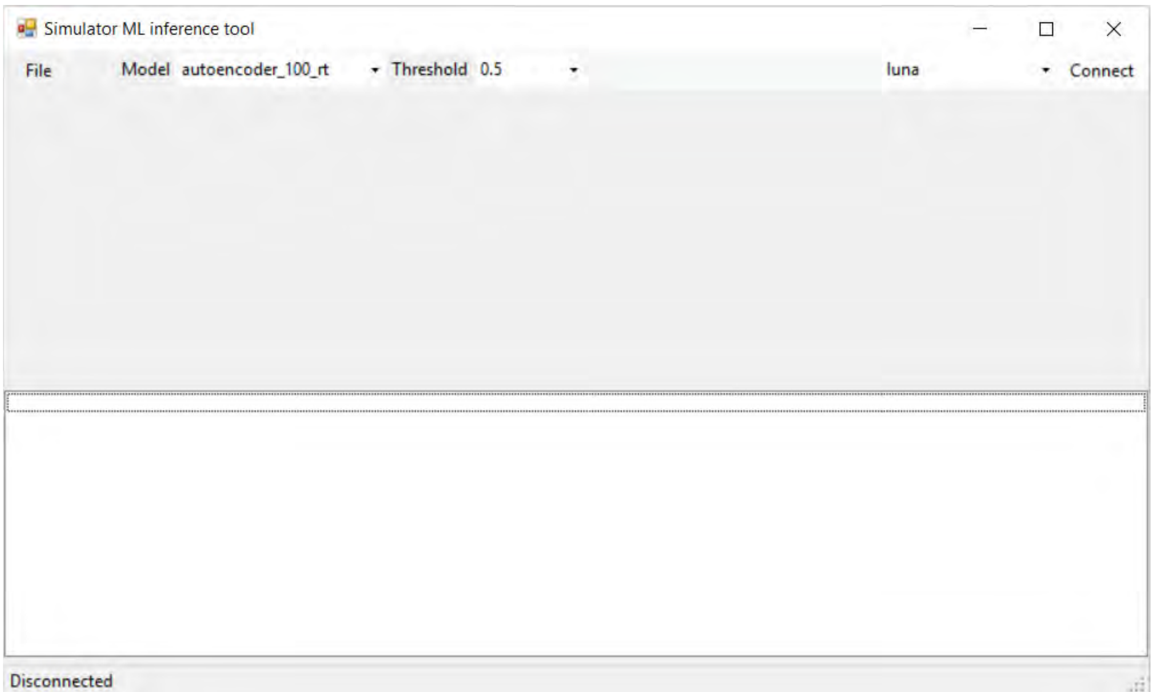

Figure 4. A snippet of the simple tool created to communicate between the simulator and ML models.

The tool allows the user to select the ML model to load and the simulator with which to connect. To load a model, a json file with the same filename as the ML model file must also be provided. The json file must contain these values:

- "friendly_model_name": the name used when displaying the model option in the dropdown box.
- "input_name": the name of the input layer of the frozen ML model.
- "output_name": the name of the output layer of the frozen ML model.
- "nmb_input_signals": the number of signals input to the ML model. This number is used when preparing the input data structure used when inferencing with the ML model. Note that this number must match the number of inputs in the input layer of the model used.
- "input_sequence_length": the length of the input sequence used when running a model inference. If the value is greater than 1, it will use a time window of that size when running model inference. Note that this number must match the sequence length used during the training of the model.

- “only_process_in”: a Boolean value to determine if only process signals are used as input or both process and alarm signals.
- “scenario”: the name of the scenario the model was trained on. This is used to load the corresponding min-max scaling data required to normalize the process signals received from the simulator.

Once loaded, the data received from the simulator signal subscription are used to run inference with the loaded ML model. The output produced by the model is then compared to the original alarm states received from the simulator. If there are discrepancies, they are considered unexpected and flagged as such. In addition, it is possible to change the threshold value used when processing the output of the ML model during runtime.

After all the signals have been processed, a message is constructed and sent to the simulator with a list of the alarms that have changed state and their new state, as predicted by the ML model. Once received, this message updates the alarm display mentioned earlier (see Figure). The display was created to facilitate comparison between the alarm lists. The tool will be made available to INL; however, modifications might be necessary to connect to INL’s simulator. ProcSee is available for use at INL as well. Minor modifications might be necessary to communicate with INL’s gPWR simulator.

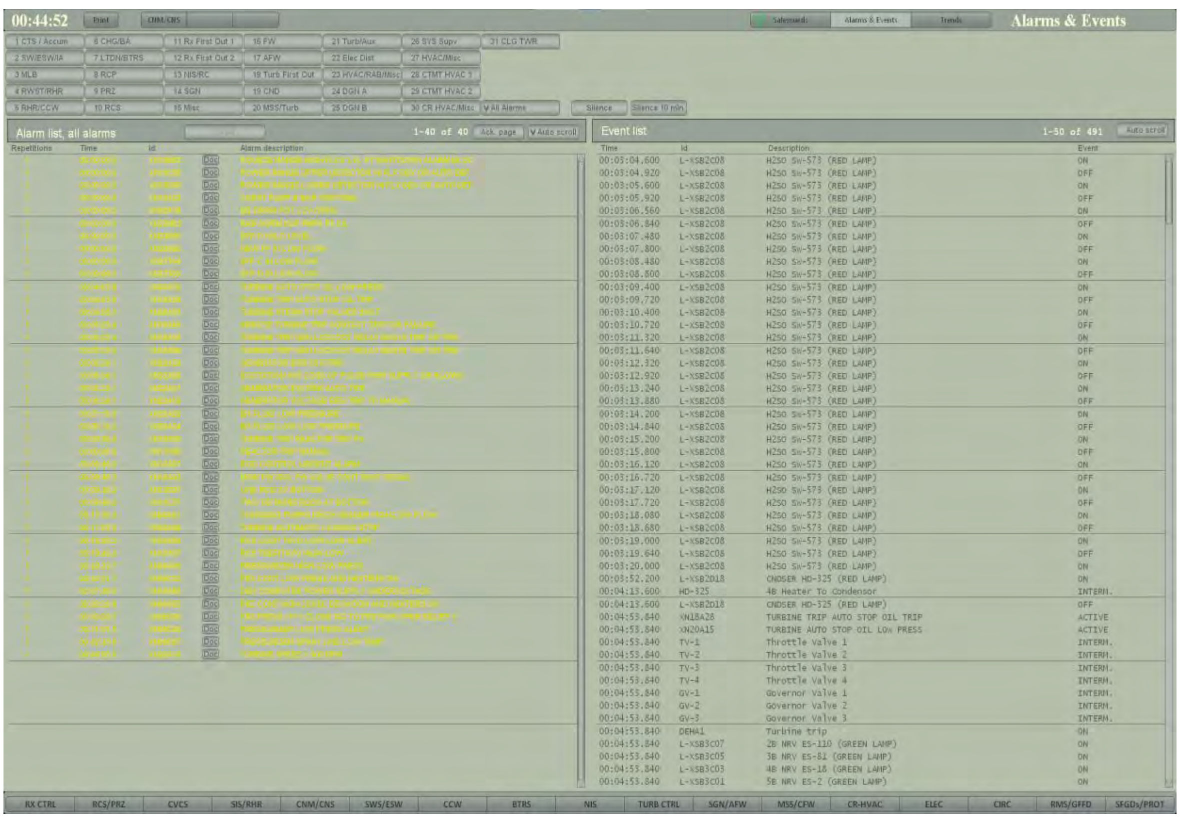

Figure 5. A screenshot of the operator display, showing both alarm lists side by side for easy comparison. The list on the left contains the original alarm list while the right list shows the ML-filtered alarm list.

The display developed for showing the alarm lists is very simple and was used only to test the ML model’s filtering capability. This does not mean that this display is the best way to visualize the results of the ML system.

6.12. Conclusions

The use of ML in the attempt to create a state-based and condition-based alarm system showed good promise. The initial tests and trials had mixed results that were dependent on which ML program was used. The ability to collect the data and train the ML program was simple and straightforward. The ability of an operator or training instructor to create the "scenarios" and identify the various states is a straightforward task. The time to set up and run the different scenarios varied from 60 and 90 minutes for Scenario 3 to 5 minutes for Scenario 4. The training of the ML program was a relatively quick process and could be done in 10 to 15 minutes for each state and further improved with increased training times. The results for Scenario 3 were positive but still had six expected alarms unidentified by ML. Additionally, ML maintained its requested state in Scenario 3 when a large steam break inside the containment building was initiated. Scenario 4 was successful in identifying expected alarms and providing the operator with information on unexpected alarms that would support future troubleshooting on the cause of the reactor trip and safety injection based on the unexpected alarms.

The research conducted under this project provided insight into the ability to use ML as a method of creating a state-based alarm system. The project showed strong positive results in the higher volume of alarms in Scenario 4. Research needs to continue to find the best ML system and most efficient method of training the ML. Including the investigation of new emerging state-of-the-art ML architectures, such as the Transformer model, which has shown great results on the processing of sequential data.

6.13. Abbreviations

AMS	alarm management/filtration system
CCW	component cooling water
CNN	convolutional neural network
DNN	deep neural network
gPWR	generic pressurized water reactor
HAMMLAB	Halden Man-Machine Laboratory
HSI	human system interface
HSSL	Human System Simulation Laboratory
IC	initial condition
IFE	Institutt for Energiteknikk
INL	Idaho National Laboratory
LSTM	long short-term memory
ML	machine learning
RCNN	recurrent convolutional neural network
RCS	reactor coolant system
RHR	residual heat removal

RNN recurrent neural network

SME subject-matter expert

6.14. References

Langstrand, J.P., Nguyen, H., and McDonald, R. (2019). Report for 2.2.1 Task 5: Develop and Document a State-Based Alarm System for a Nuclear Power Plant Control Room Using Machine Learning, INL/EXT-19-55368. Idaho Falls: Idaho National Laboratory.

6.15. Figures Section

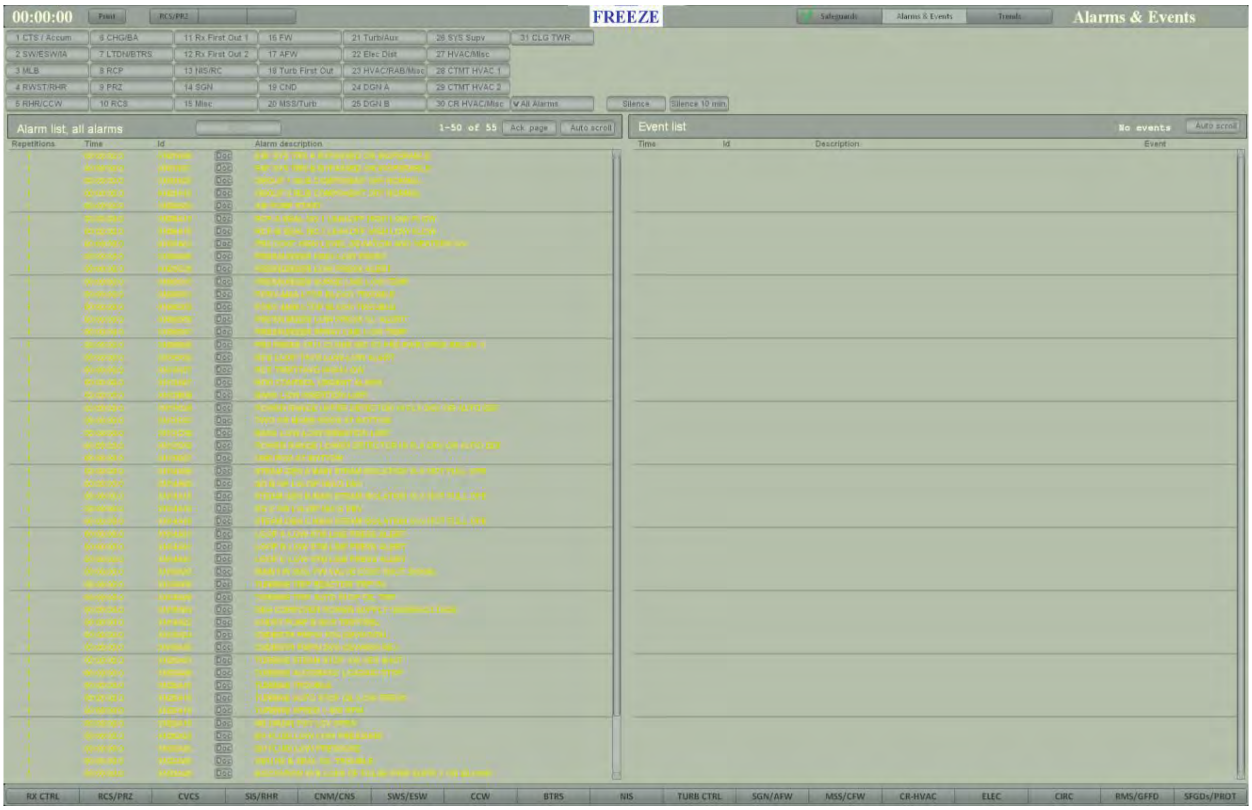

Figure 6. Initial alarm display for Scenario 1, 55 alarms.

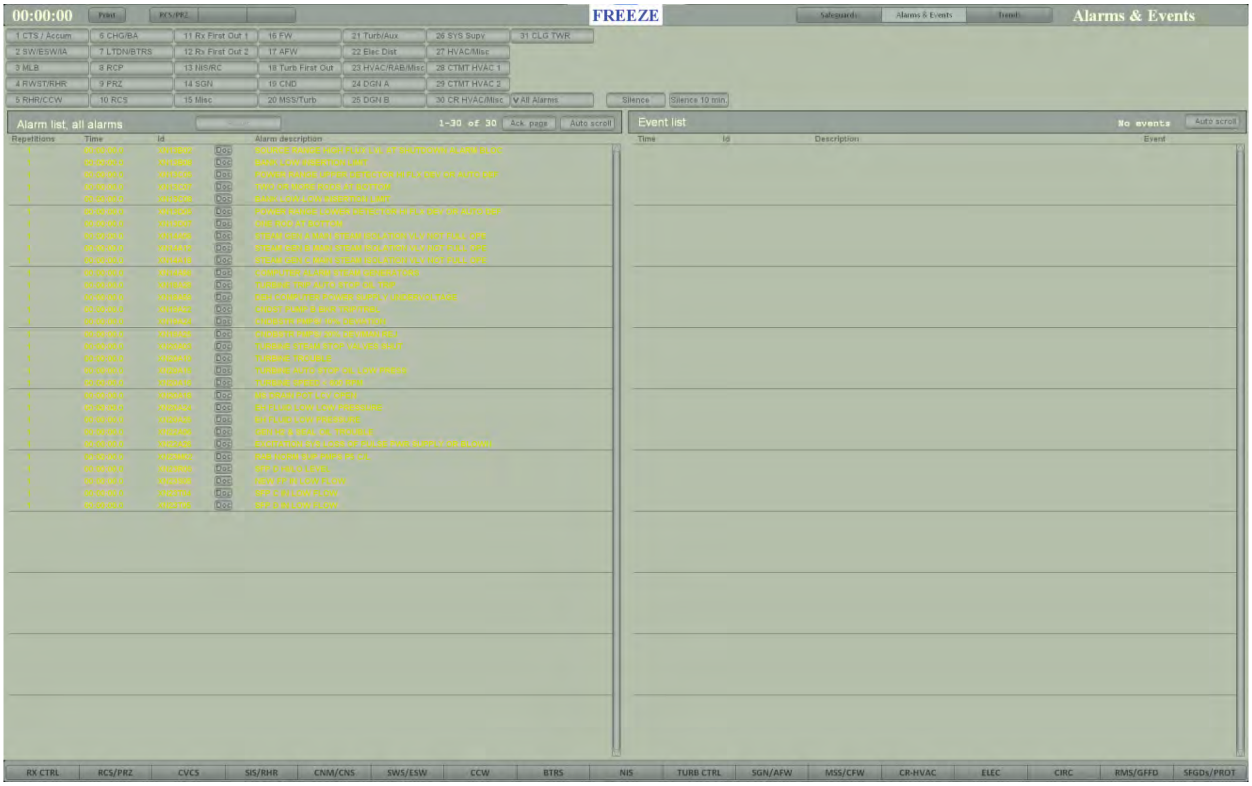

Figure 7. Initial alarm display for Scenario 2, 30 alarms.

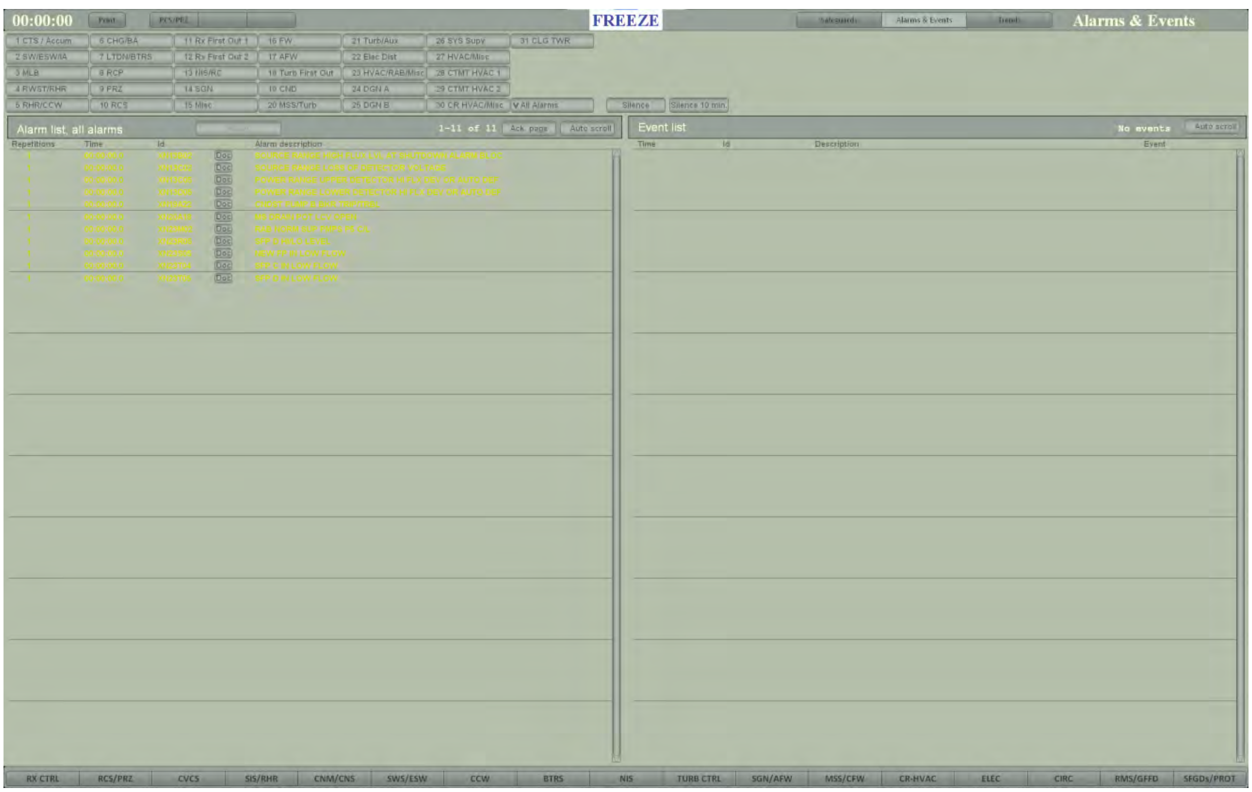

Figure 8. Initial alarm display for Scenario 3, 11 alarms.

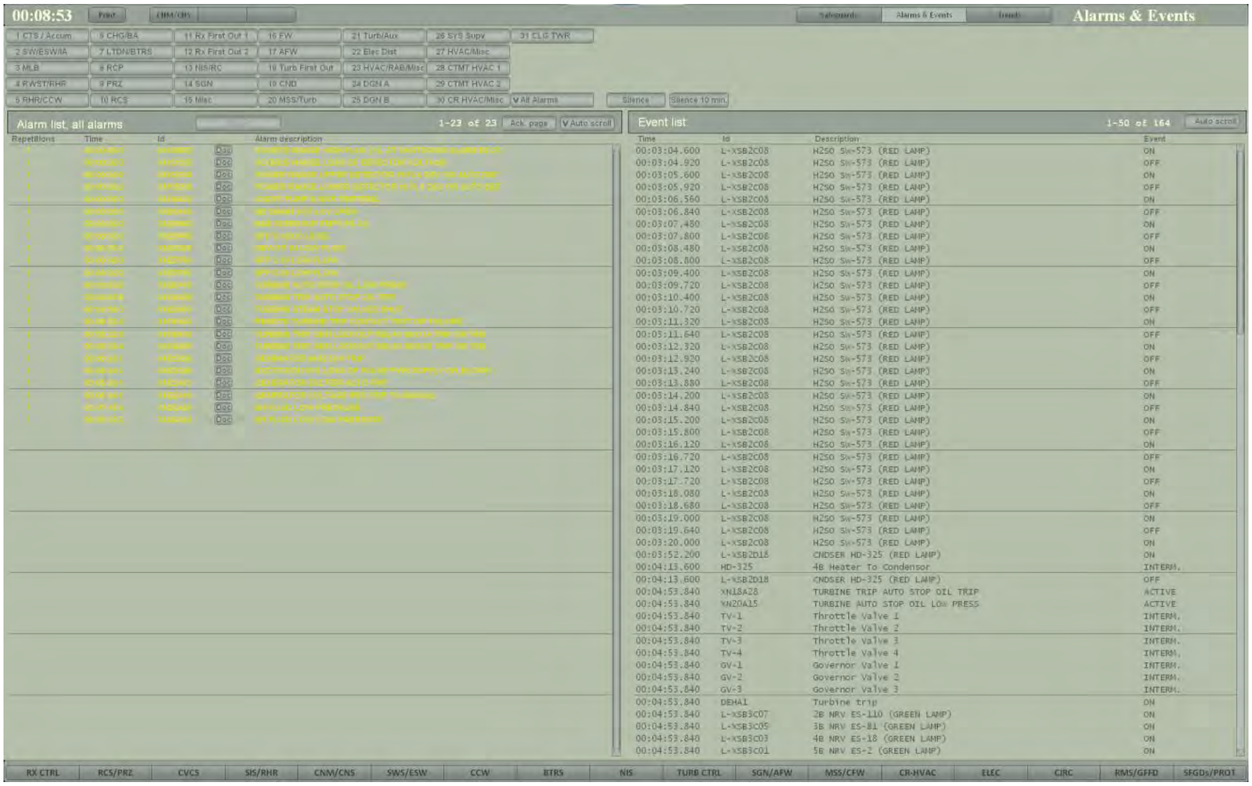

Figure 9. Scenario 3 after a turbine trip, an additional 12 alarms for a total of 23.

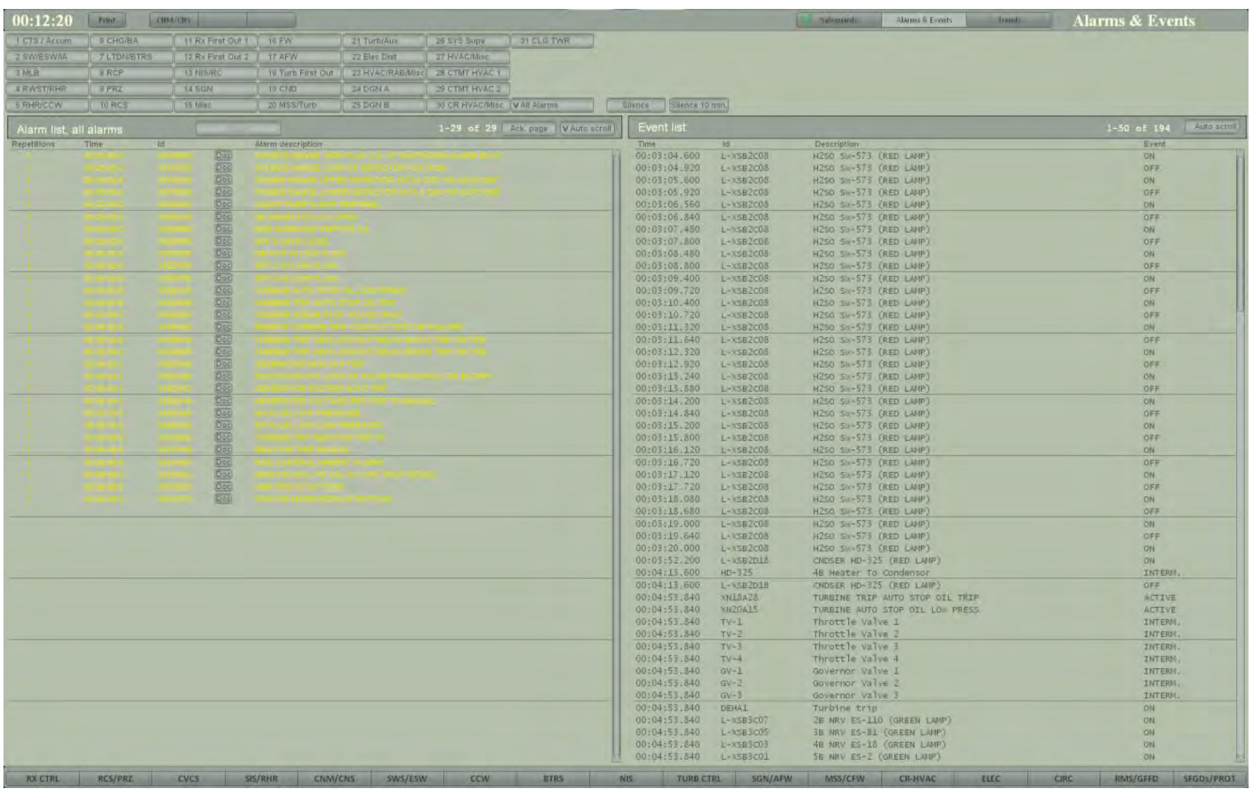

Figure 10. Scenario 3 after a reactor trip, an additional six alarms for total of 29.

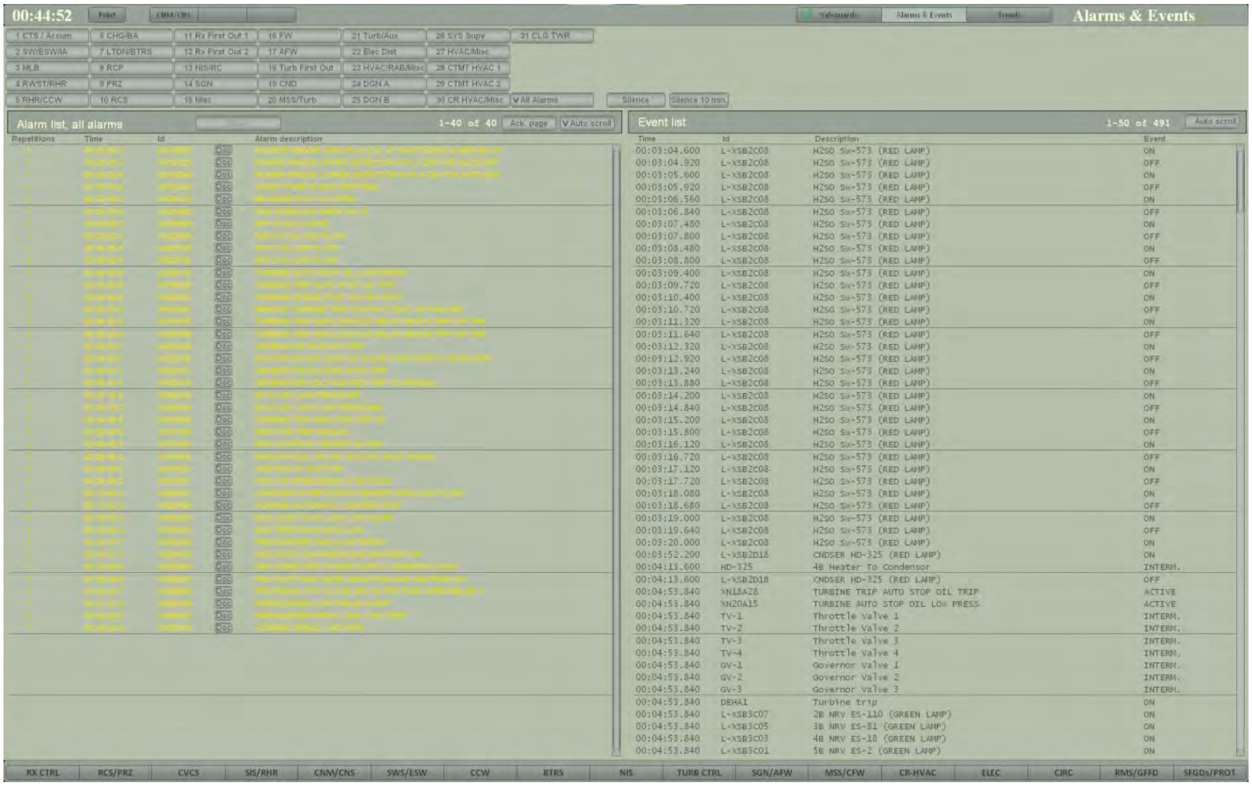

Figure 11. Scenario 3 end of scenario, an additional 11 alarms for a total of 40 alarms.

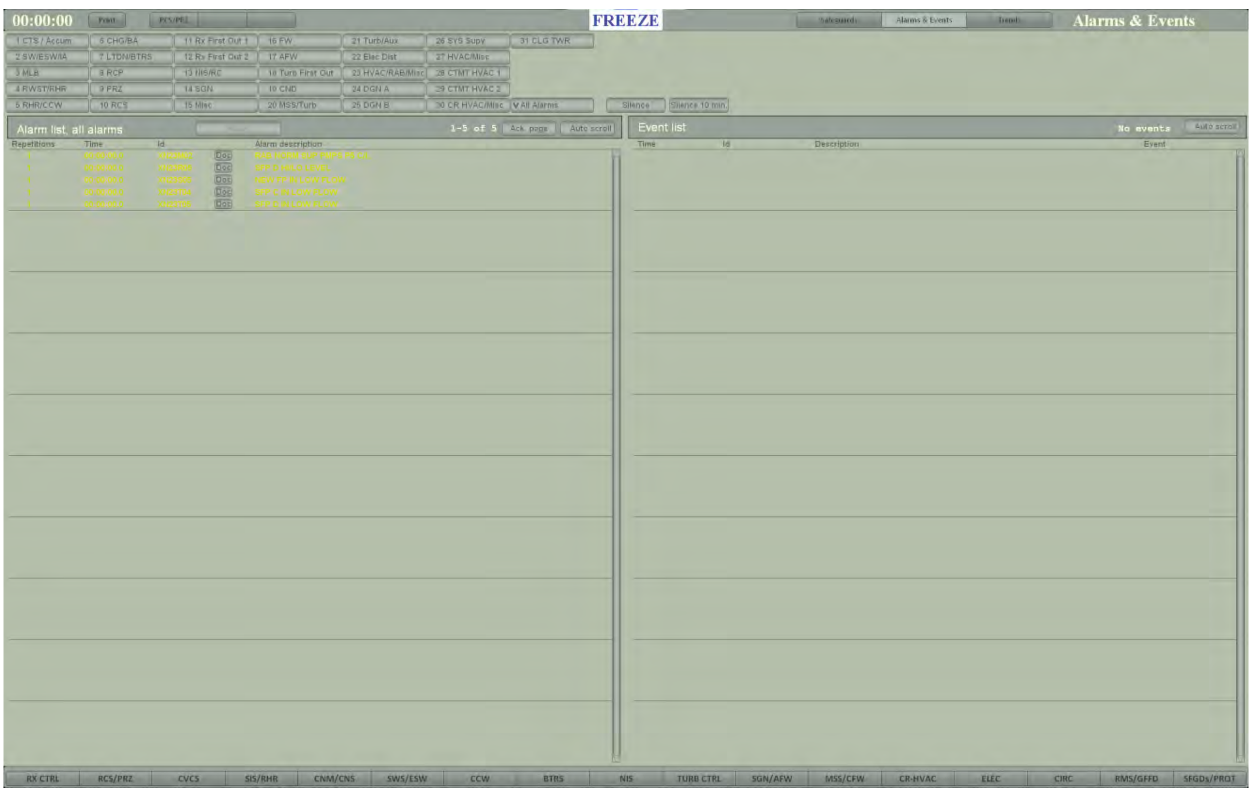

Figure 12. Initial alarm display for Scenario 4, five alarms.

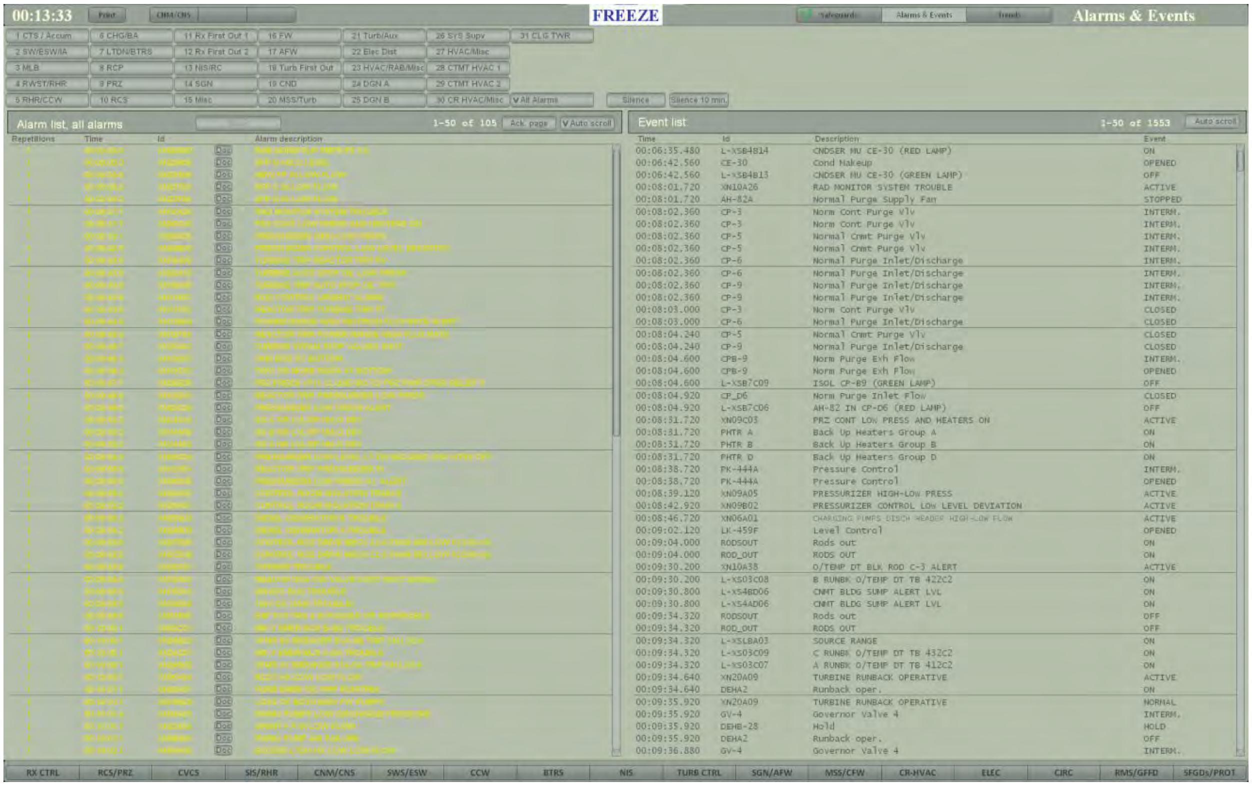

Figure 13.1. Alarm display for Scenario 4 after trip and SI, alarms 1 to 50.

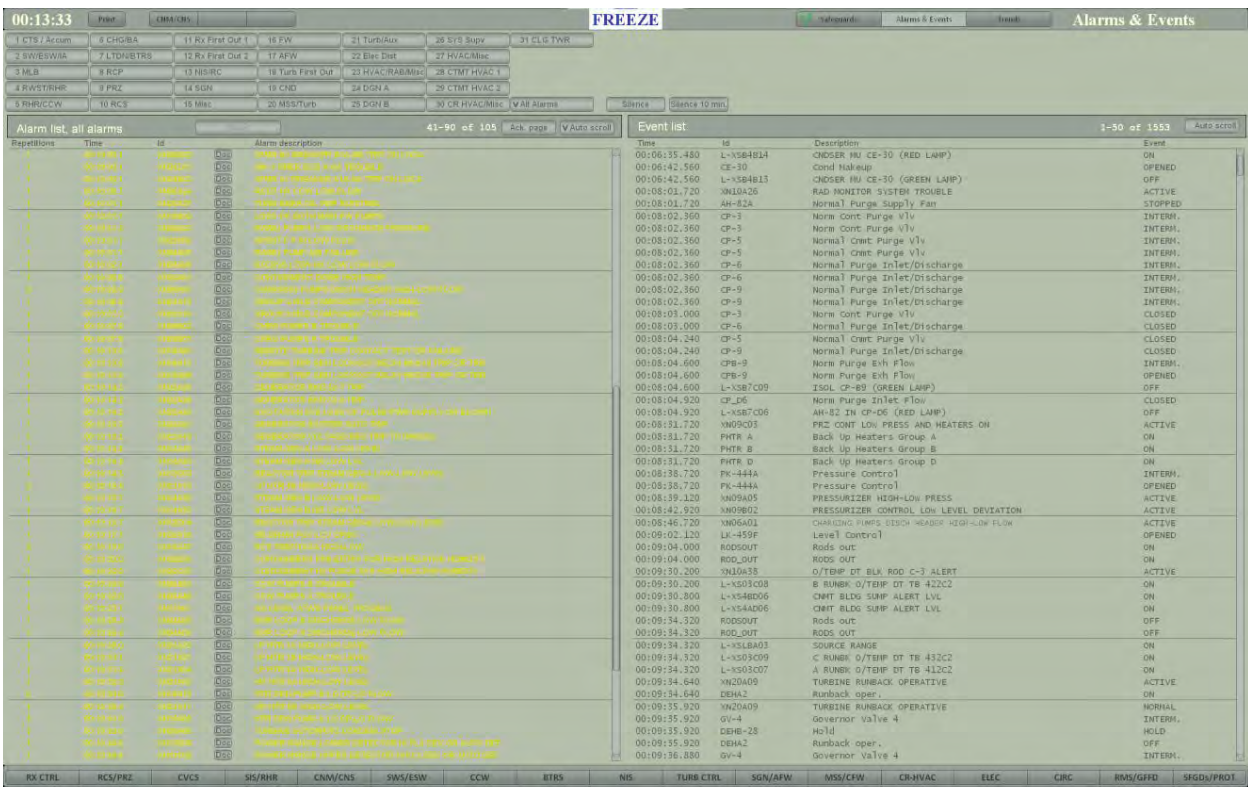

Figure 13.2 Alarm display for Scenario 4 after trip and SI, alarms 41 to 90.

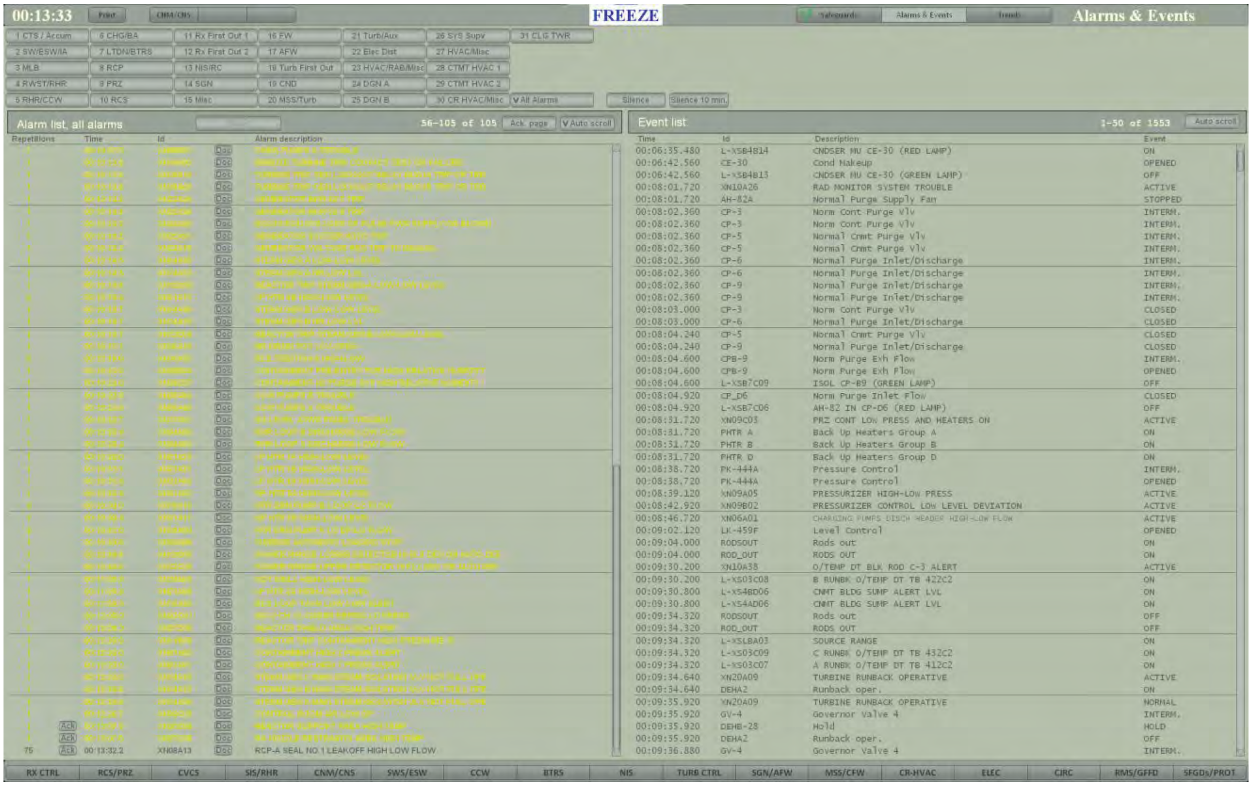

Figure 13.3. Alarm display for Scenario 4 after trip and SI, alarms 56 to 105.

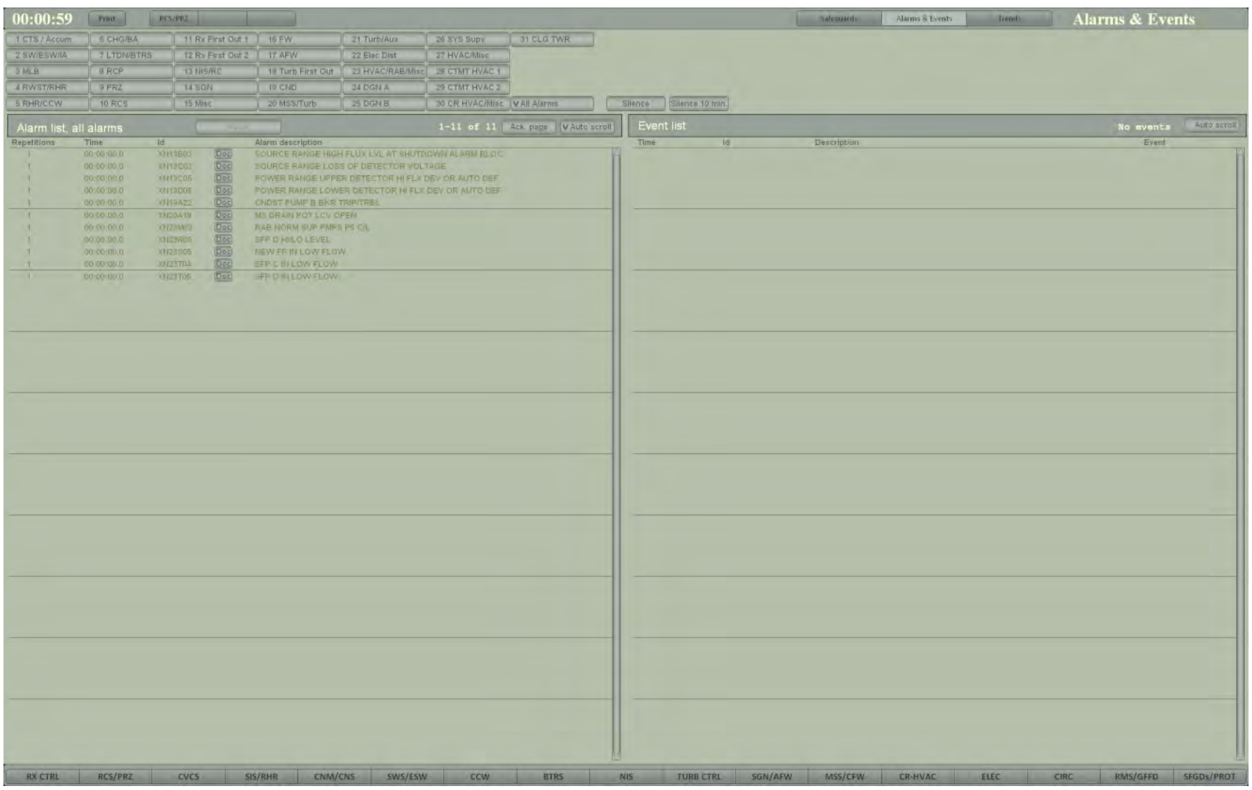

Figure 14. Alarm display for Scenario 3 initial condition, all alarms identified by ML as expected.

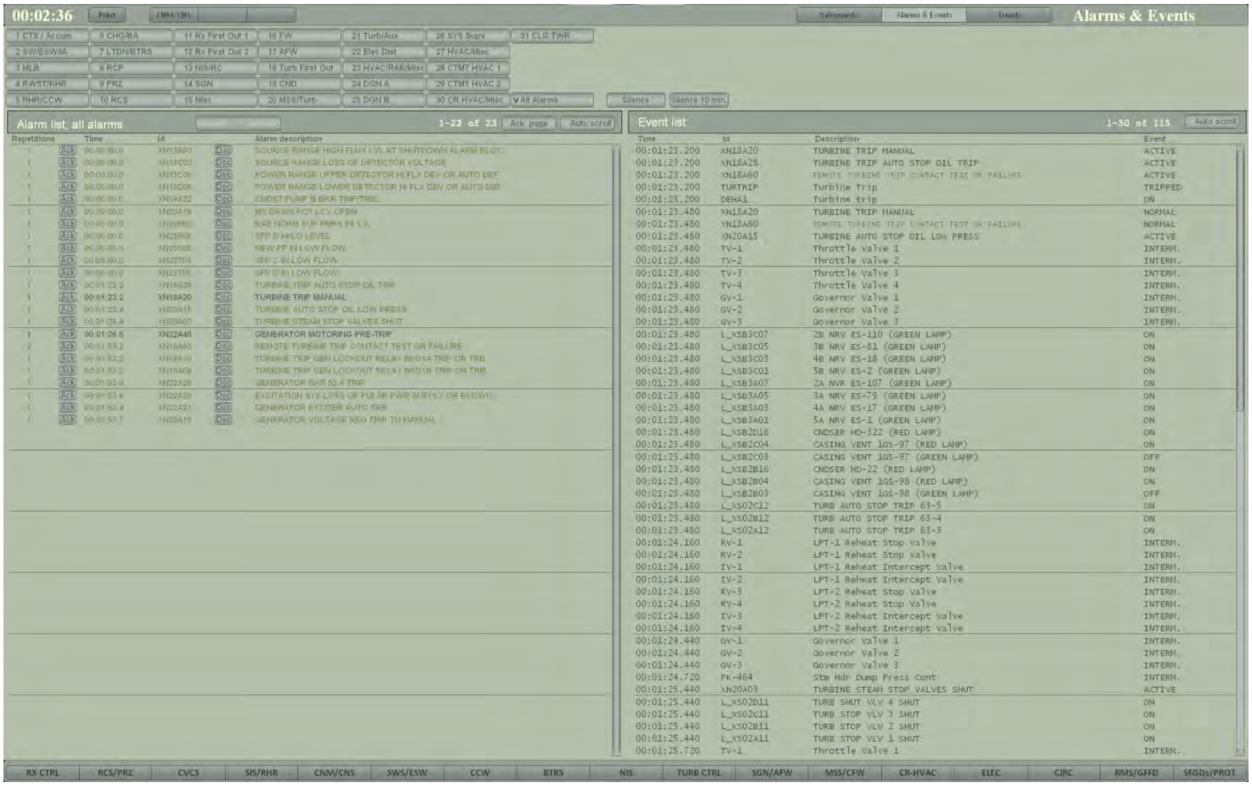

Figure 15. Alarm display for Scenario 3 after a turbine trip, all alarms identified by ML as expected.

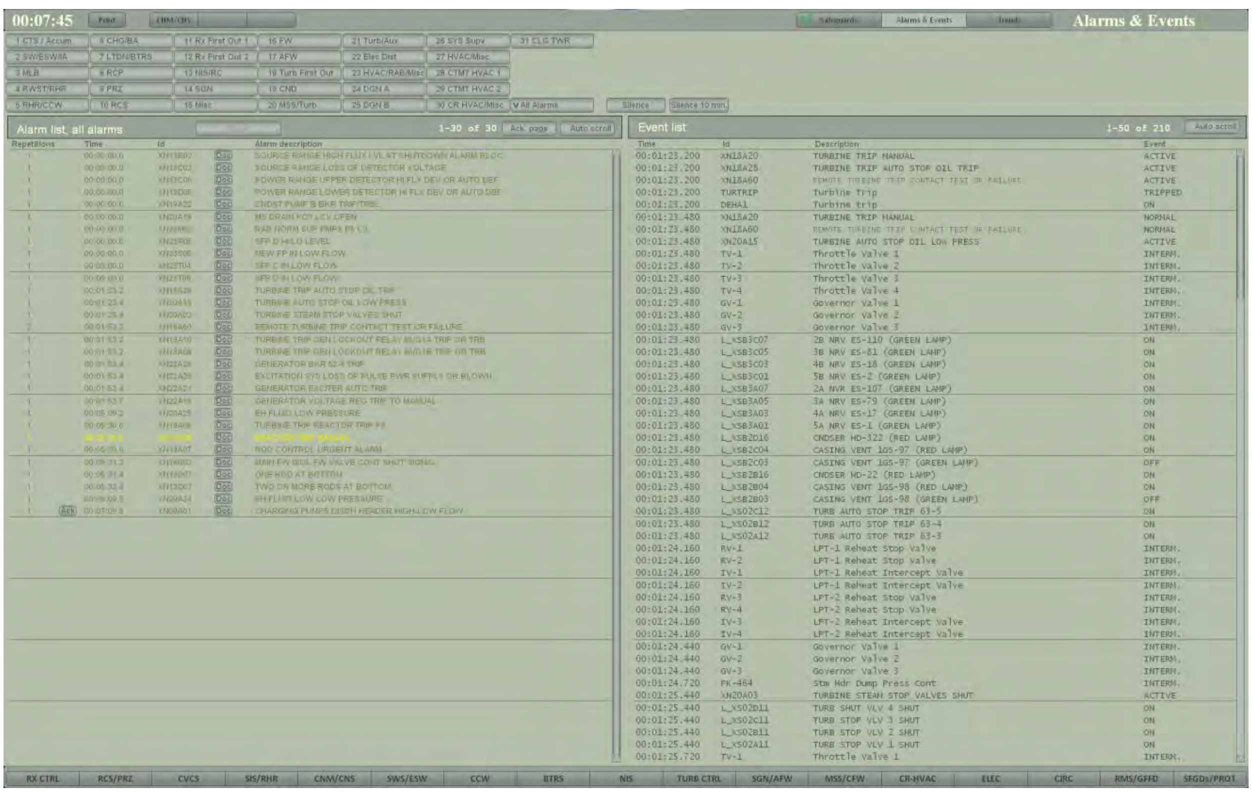

Figure 16. Alarm display for Scenario 3 after a reactor trip, all but one alarm identified by ML as expected.

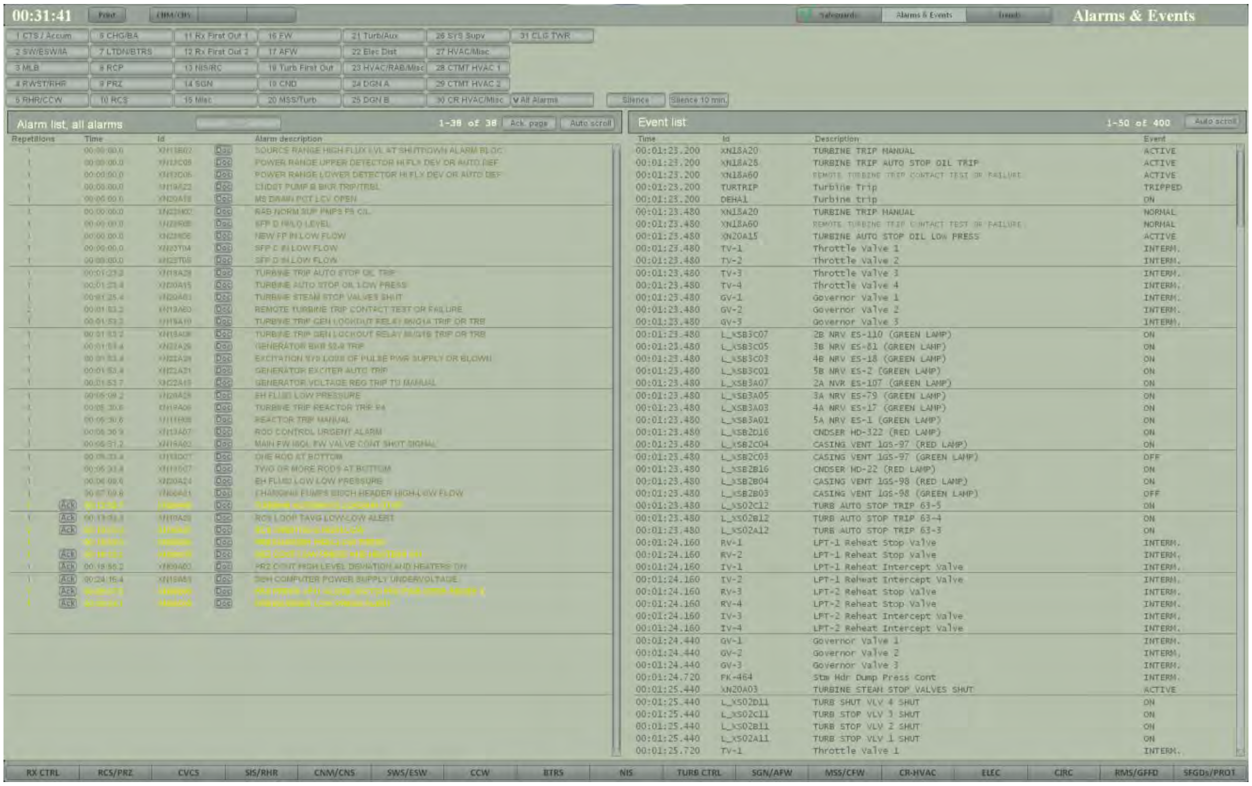

Figure 17. Alarm display for Scenario 3 after cooldown, all but six identified by ML as expected.

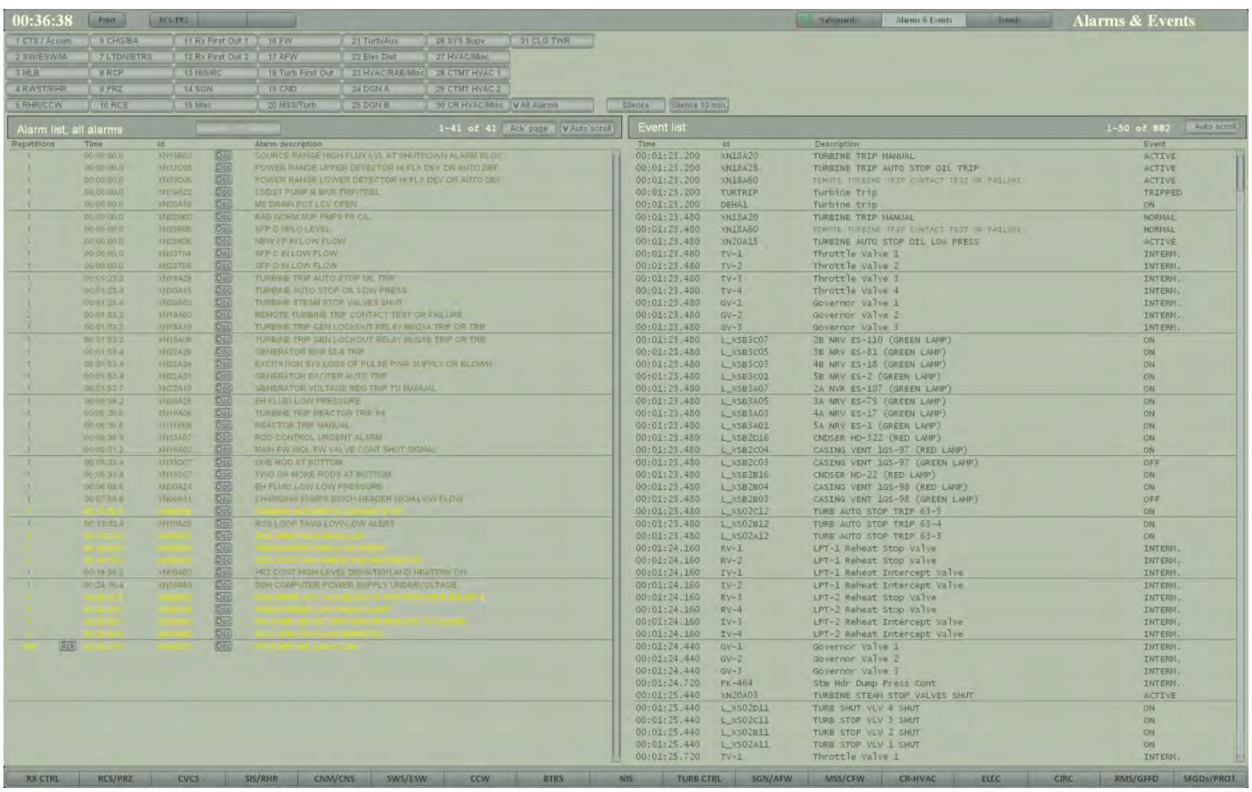

Figure 18. Alarm display for Scenario 3 with the faults inserted, none of the faults identified by ML as expected. No change in the six previous unidentified alarms.

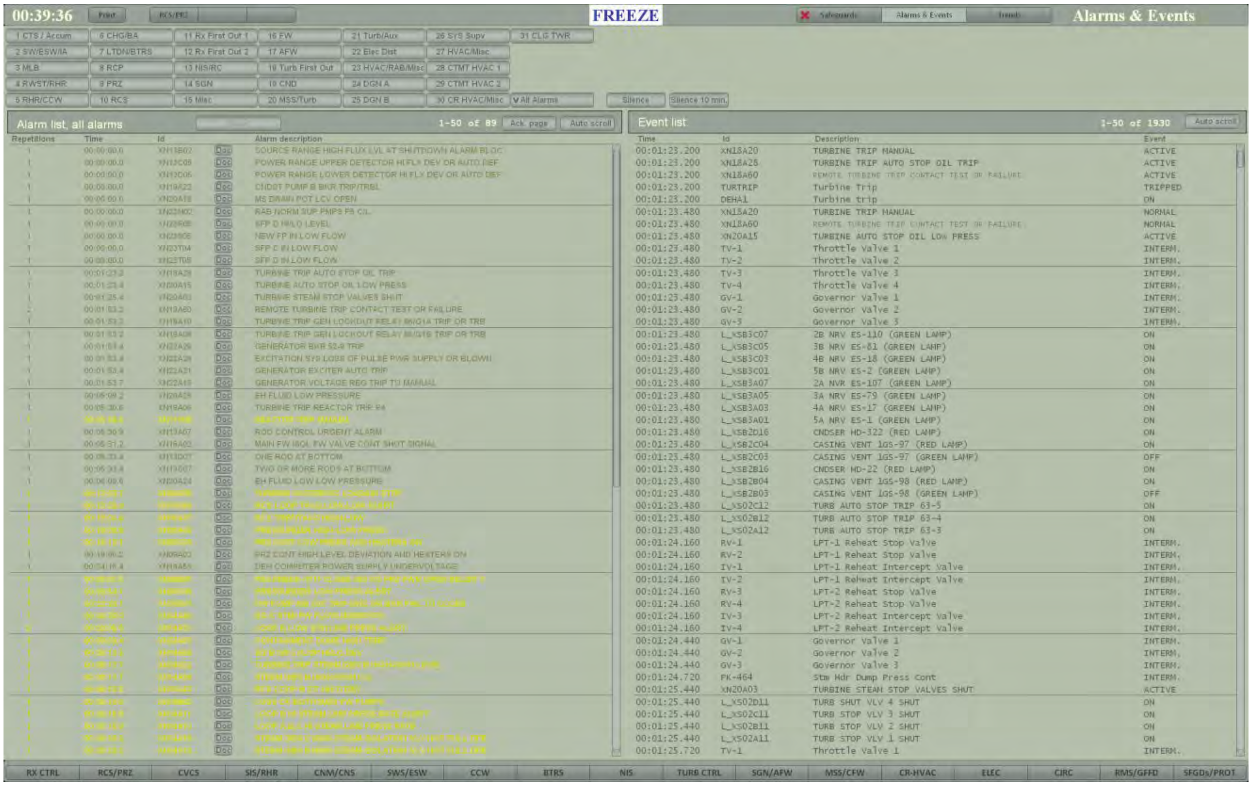

Figure 19.1. Alarm display for Scenario 3 after the final fault inserted alarms 1 to 50.

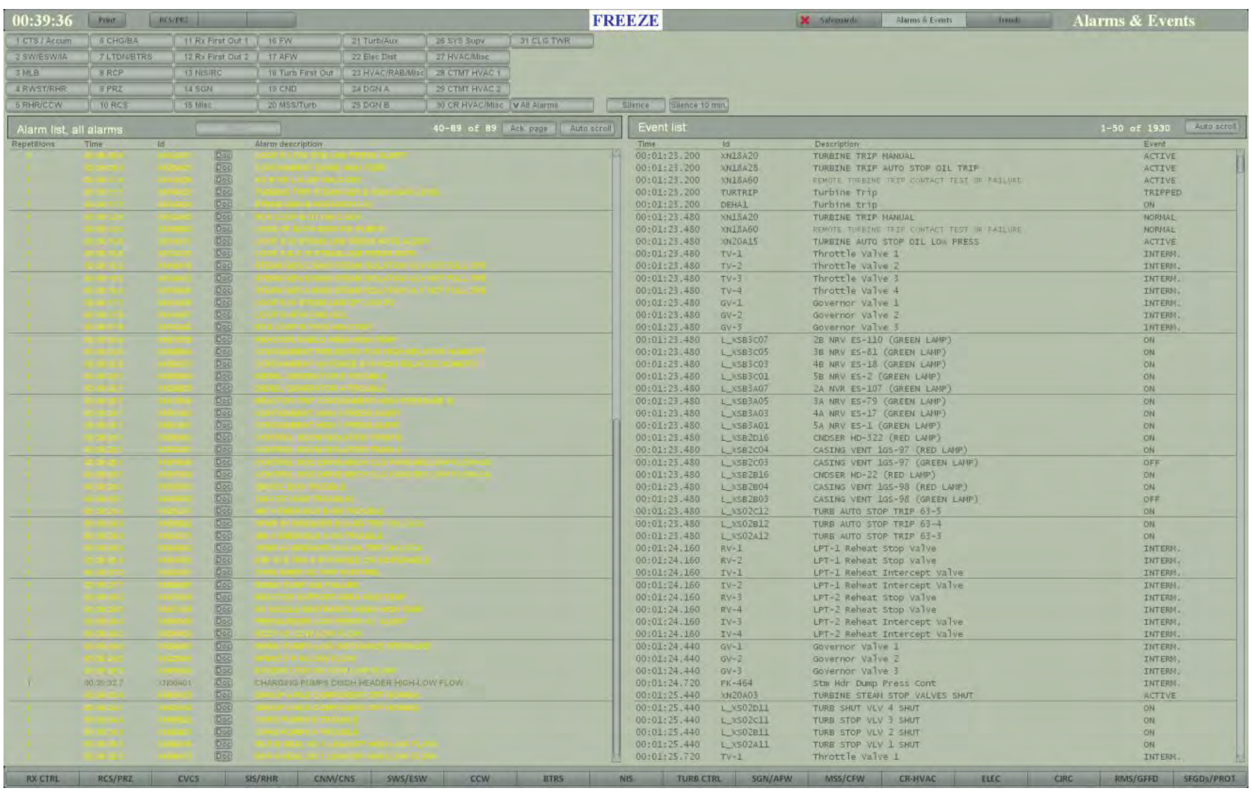

Figure 19.2. Alarm display for Scenario 3 after the final fault inserted alarms 40 to 89.

Figure 20.1. Alarm display for Scenario 4 after the final fault inserted alarms 1 to 50.

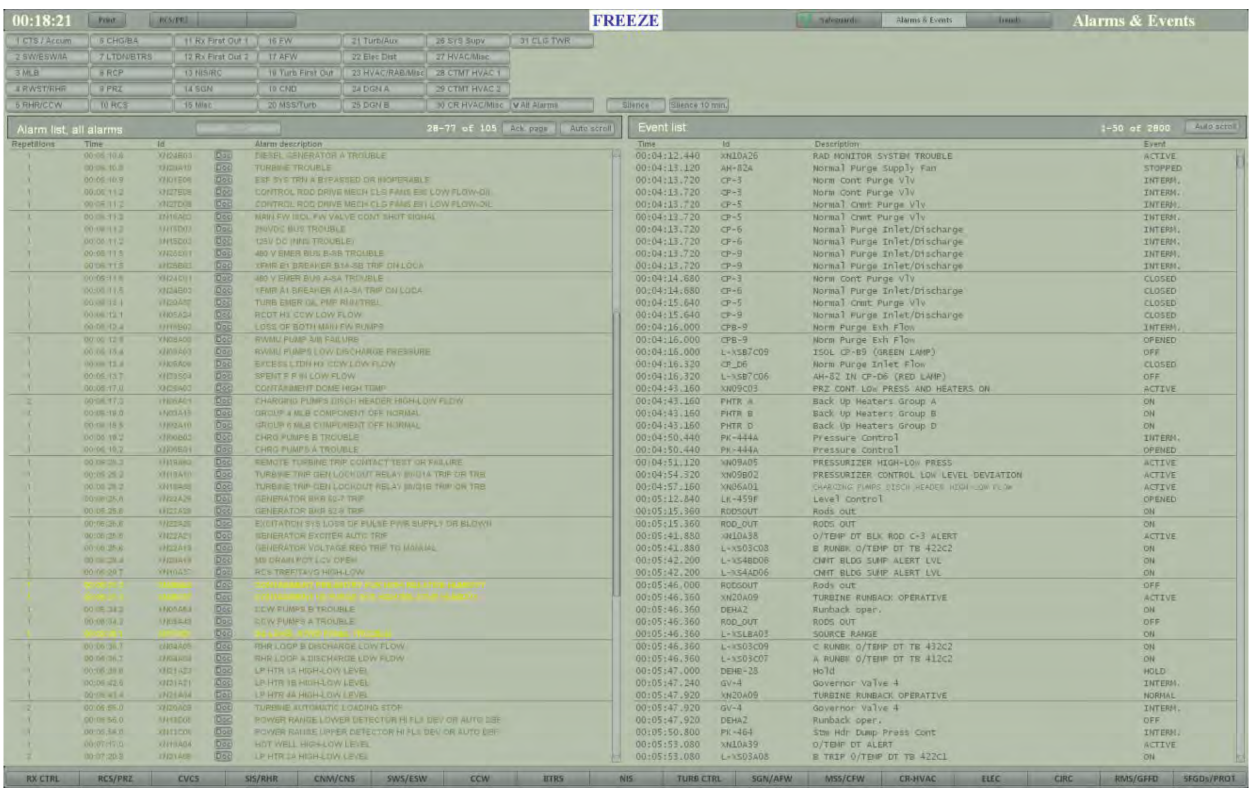

Figure 20.2. Alarm display for Scenario 4 after the final fault inserted alarms 28 to 77.

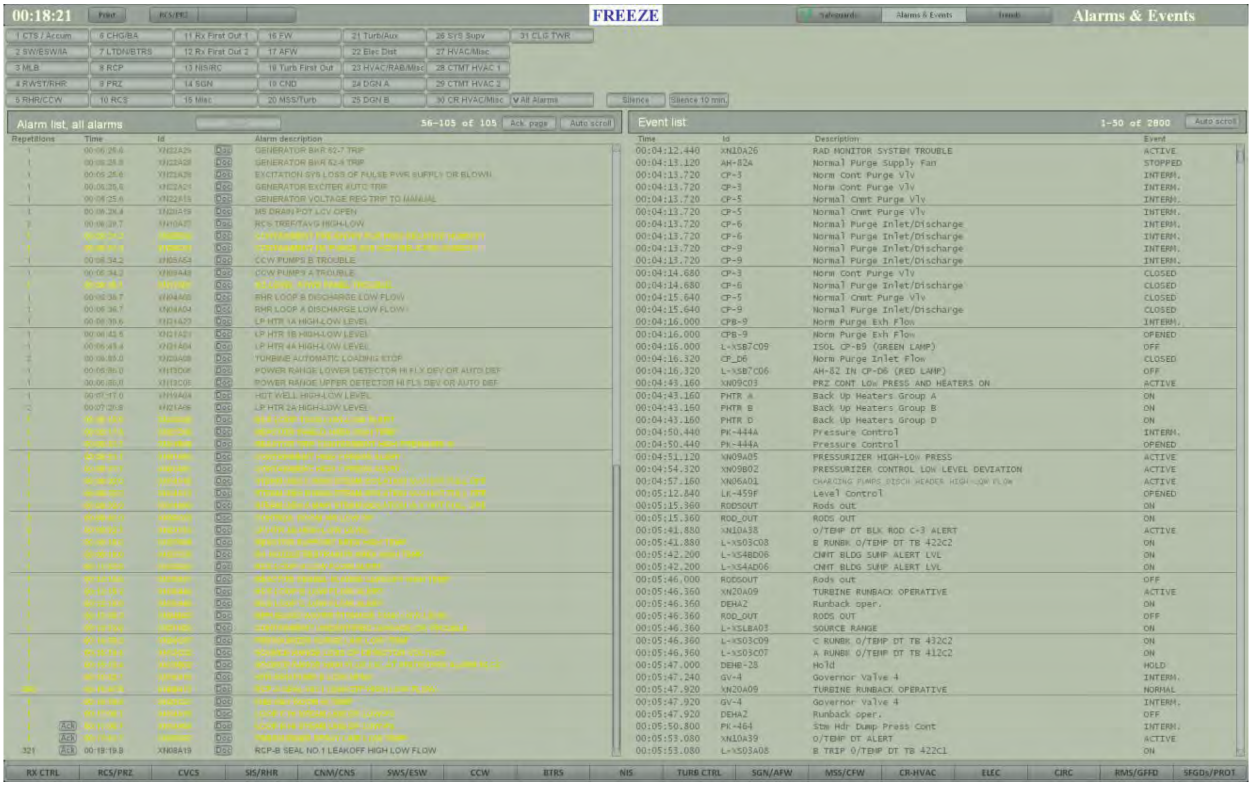

Figure 20.3. Alarm display for Scenario 4 after the final fault inserted alarms 56 to 105.

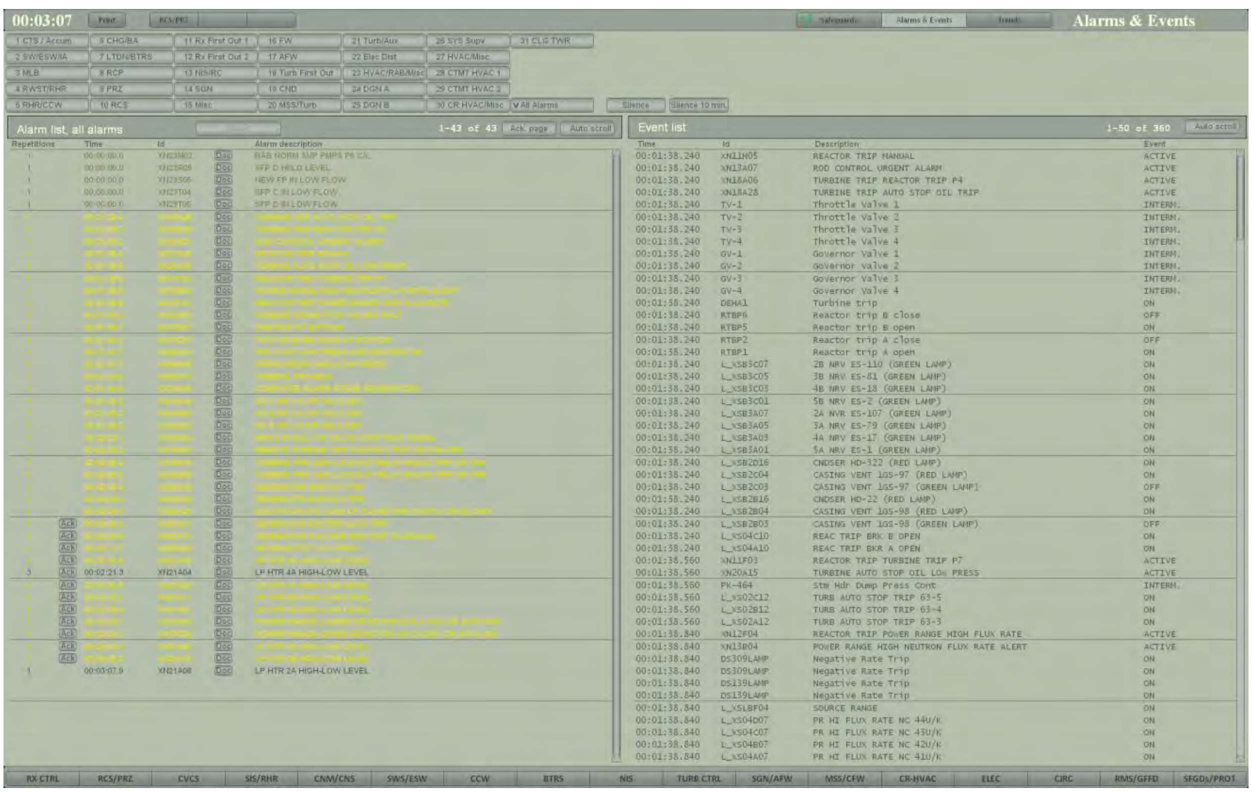

Figure 21. Alarm display for the Scenario 4 first test first run. The initial alarms were identified, but none of the other expected alarms were identified after an Rx trip. Two cleared alarms are present.

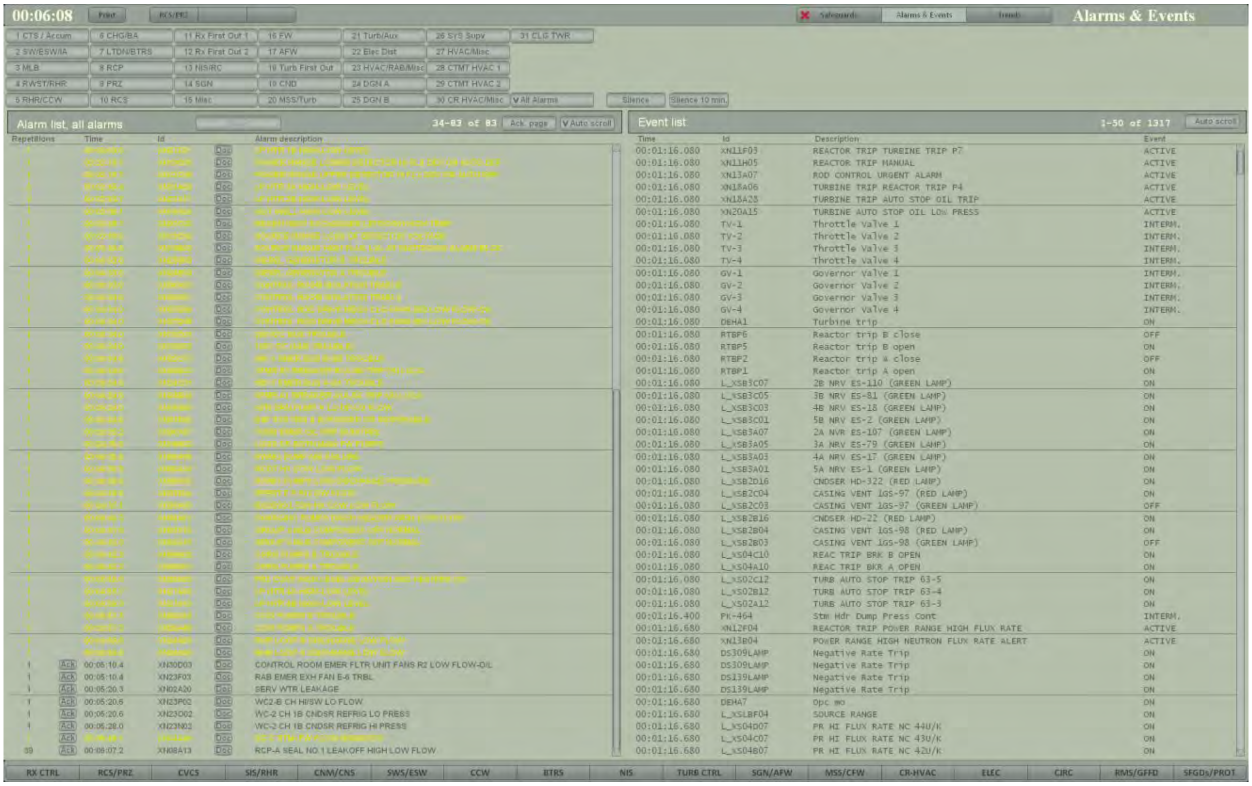

Figure 22. Alarm display for the Scenario 4 first test first run, after SI initiated alarms 34 to 83. No expected alarms are identified. Seven cleared alarms are present.

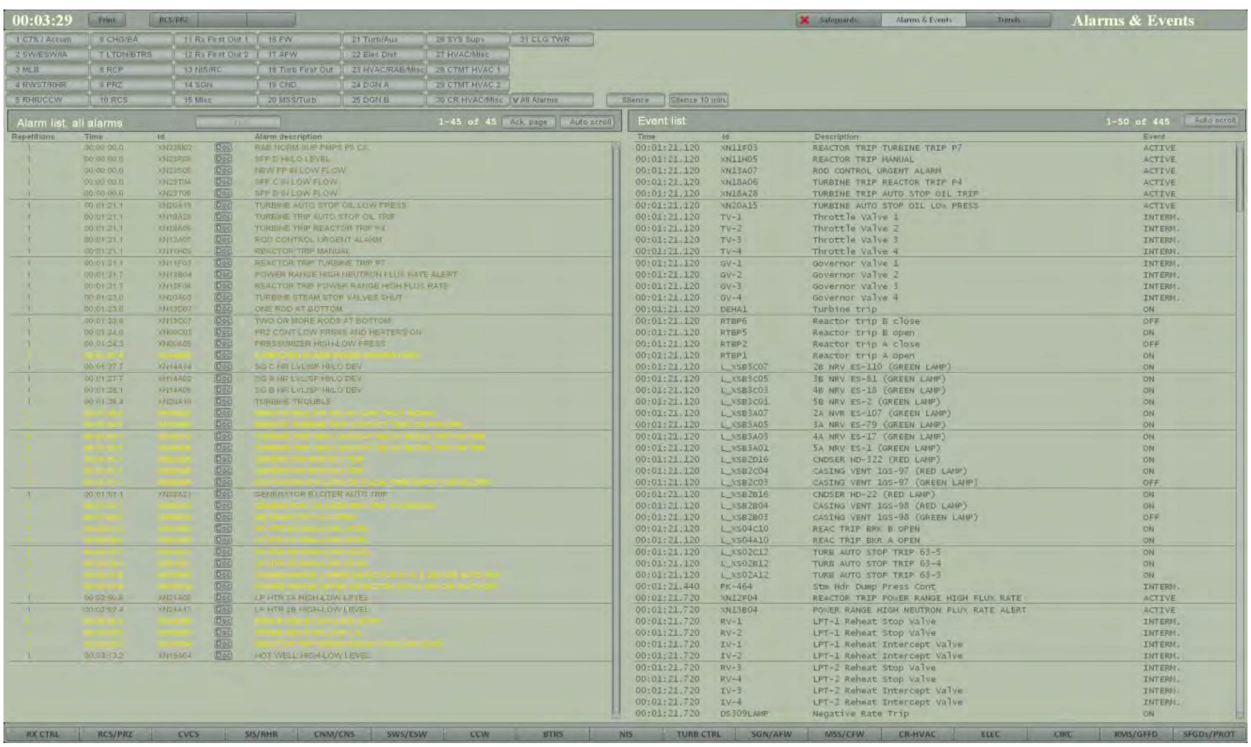

Figure 23. Alarm display for the Scenario 4 first test second run, after an Rx trip; 26 identified unexpected alarms, 19 unidentified expected alarms.

Figure 24.2. Alarm display for the Scenario 4 first test second run, after SI initiated alarms 1 to 50. Forty two of the 80 expected alarms are identified; 38 expected alarms not identified.

Figure 24.2. Alarm display for the Scenario 4 first test second run, after SI initiated alarms 31 to 80. Forty two of the 80 expected alarms are identified; 38 expected alarms not identified.

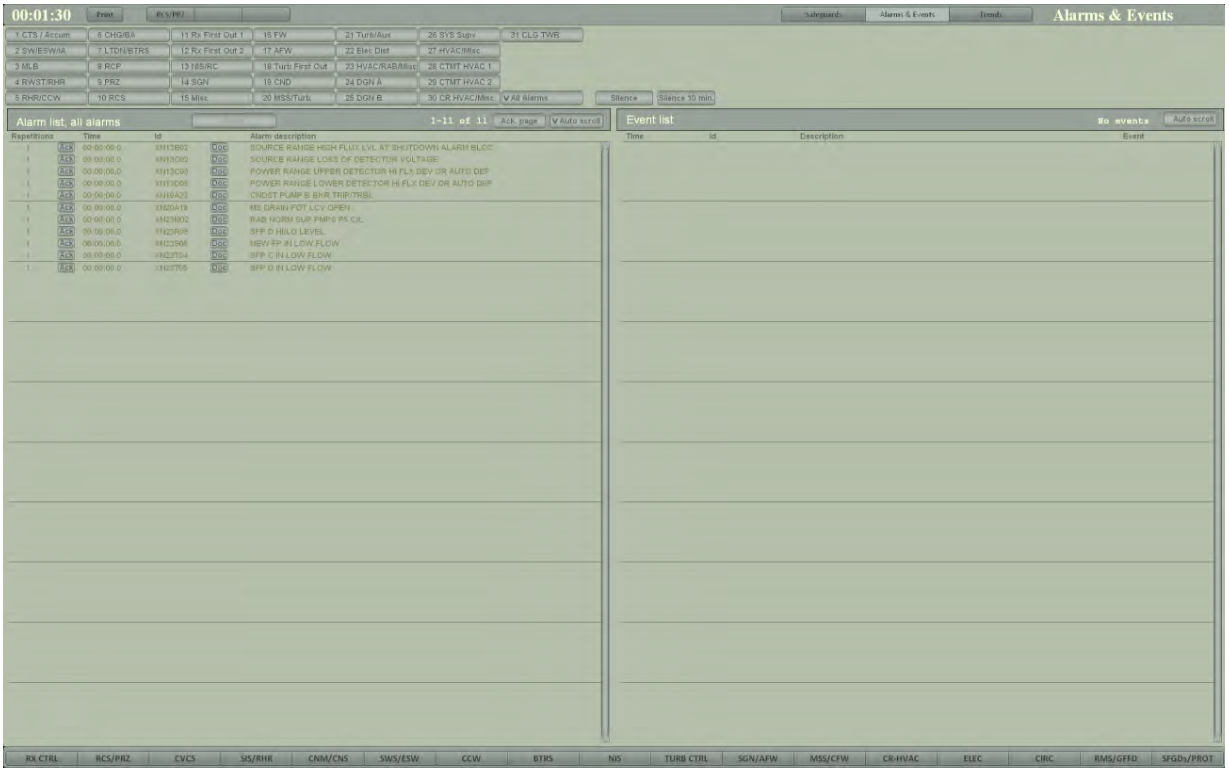

Figure 25. Alarm display for the Scenario 3 first test. Initial alarm screen, all expected alarms identified.

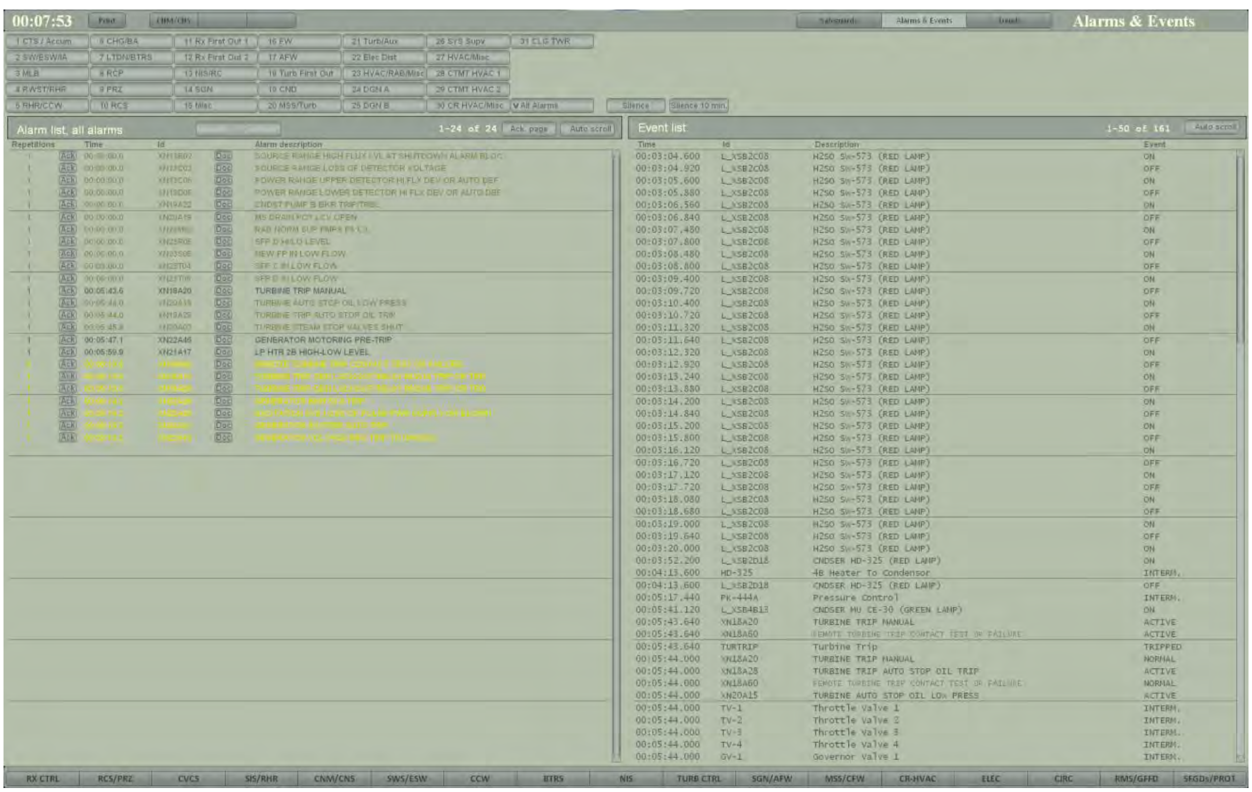

Figure 26. Alarm display for the Scenario 3 first test, after a turbine trip. ML identified three additional expected alarms, missed seven expected alarms, and includes three cleared alarms.

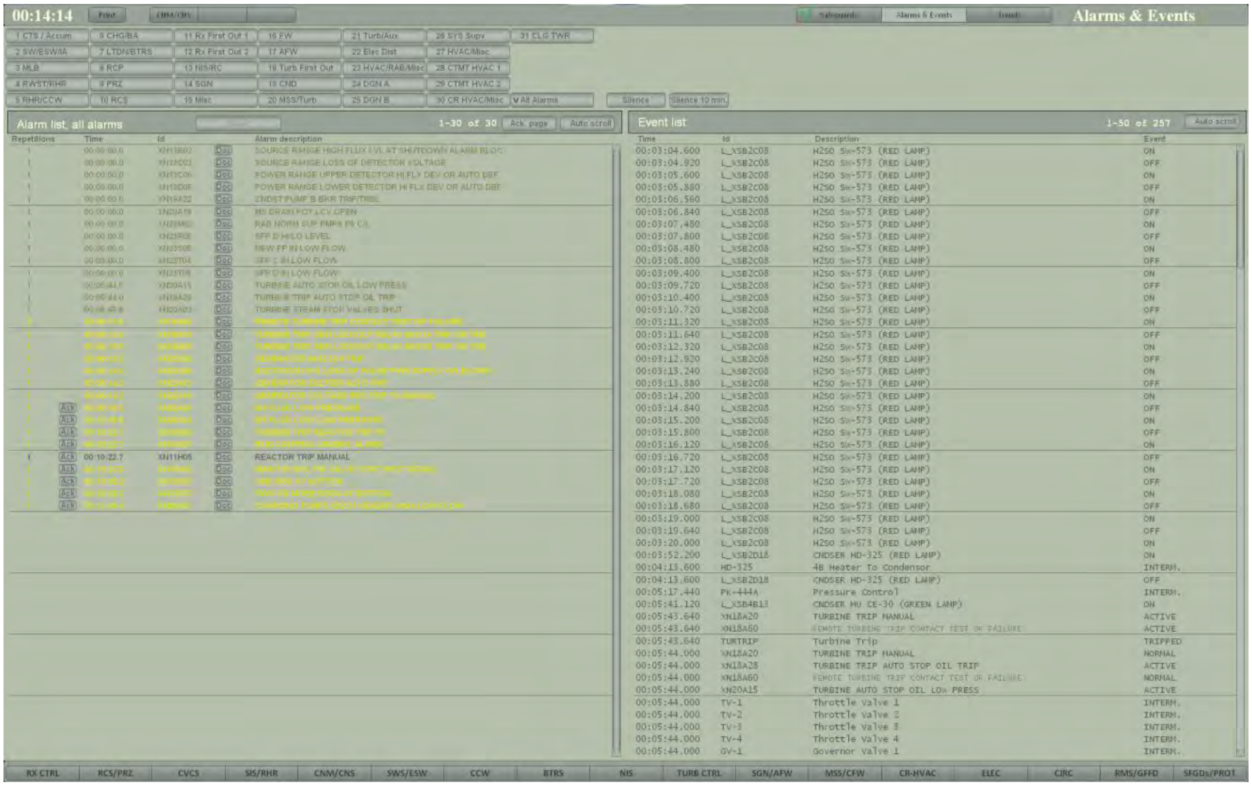

Figure 27. Alarm display for the Scenario 3 first test, after an Rx trip. Six additional alarms were not identified, and there was no change in identified alarms. One cleared alarm is included.

6.16. Anonymous Peer Review to Validate the Efficacy of Using ML to Develop State-Based Alarm Solutions

In FY19, LWRS Program researchers worked with staff from the Halden Reactor Project (IFE) to develop a state-based alarm system using machine learning. This initial proof of concept R&D was documented in Langstrand, Nguyen, and McDonald (2019). Building on this work, IFE and INL conducted the R&D documented in this report to further develop this approach.

Part of the scope of this new R&D activity is to have the state-based alarm system developed using ML peer reviewed by subject matter experts (SMEs) who are highly familiar with the gPWR simulator to validate the effectiveness of using ML to develop state-based alarm solutions. As such, two SMEs, who were former licensed operators of the commercial nuclear power plant upon which the gPWR is modelled, were recruited to peer review this report and the results of the ML testing.

In synthesizing the SME's comments, a few general themes appear in their review. An overall theme from their comments is that the research results thus far show some promise, but that there is still a considerable amount of additional work needed in order to say more definitively whether this approach has potential or not. Said differently, a key inference from synthesizing the SME's comments is that they found it difficult to say whether or not Halden did a good job in developing a state-based alarm system because the report did not have enough detail to render a verdict one way or the other. For example, when it came to answering a very direct question about the approach, such as, "Did the ML correctly suppress expected alarms, and correctly 'let through' unexpected alarms?" the SMEs demurred in their response.

A second theme arising from the SME's comments is that there appeared to be cases where Halden's test design was off the mark. In some cases, the SMEs commented that the scenario tested and/or the ML generated output were not of much use to them. While this negative feedback could be taken as harsh, it is hoped that Halden will view this feedback as suggestions for how to improve their development efforts. In fact, a third theme from the SME's comments consists of explicit suggestions from the SMEs on what Halden should consider doing as they continue to perform this research.

The SME's specific comments are as follows:

a. **6.2.3 Scenario 3**. Figures 8, 9, and 10. A normal shutdown with no faults. All three screens were in bright color and most of the alarms were normal for this scenario. I thought the ML would take all alarms and output in the bolder bright color the abnormal alarms for the plant conditions. If I were shutting down the plant these three screens would be of no use to me. With my understanding this scenario would be a training scenario for normal alarms.

b. **6.2.4 Scenario 4.** I'm not sure what to make of it. Is it a training scenario for ML? Figures 13-1 through 13-3 are all in bright color except for one alarm on the last page. This statement is at the end of this scenario "the alarms identified as not expected will standout and give the control room staff a better understanding for the cause of the trip or safety injection." I don't see that with these 3 alarm screens.

c. **6.3.1 Scenario 3 with faults**. Figure 17 should be before any faults are inserted. It shows 6 alarms in bright yellow that are normal for the plant conditions which are shutdown and cooled down. Figures 19.1 and 19.2, show the steam line break inside containment on loop B. There is still a large volume of alarms (89) to sort through which would not quickly point me to the cause of the abnormal plant condition. The following are examples of what would help. Steam generator B high-high level and the reason, the lowering pressure in the steam generator causes swell in the steam generator as it rapidly boils due to the reduced pressure. Loop B AFW line isolation because it automatically isolated the B steam generator from the other steam generators. Containment High 1(2) pressure alert. High temperatures in Containment. The ML has to distinguish important abnormal alarms from unimportant abnormal alarms to be useful in quickly identifying a cause. For example, the main steam isolation valves going closed is abnormal but is a result of high containment pressure due to the steam break. Checking the main steam isolation valves went shut would be a follow-up action for me to verify but doesn't help me identify the cause of the event.

d. **6.3.2 Scenario 4.** The ramp time for the LOCA would cause a few initial alarms as shown in Figure 20.1 and would direct my attention to the whole plant response as the failure ramps in and I would take action. Since the fault is ramped in, I would have time to respond to the plant. I would see pressurizer level lowering, pressure lowering, and I would take manual action to trip the Reactor prior to an automatic action. I would then enter the emergency procedures. In this scenario ML would not help me a lot because it doesn't pinpoint anything specific to me. That is because after the first alarms of lowering pressure and level deviations my attention would be focused on the plant and not so much on alarm screens. The plant is showing me where its headed. Useful alarms in this case are pressurizer level, pressure and containment pressure. All of those point to a break inside containment. In this scenario ML wouldn't be that useful but if the scenario were a power operated relief valve opening the abnormal condition immediately showing up on the screen would tell me to direct the operator to close the isolation valve for that PORV. If ML were implemented and the Operator immediately looked to the screen for something to pop up from ML it could be useful to them.

e. Final comments.

   i. When the Reactor trips the crew goes into emergency procedures. The ML alarm screen would have to be pretty specific with its alarms to be useful at this point. I wonder if ML might be most useful during scenarios that do not put the plant into a Reactor trip.

   ii. How would you train ML? Use the plant specific simulator?

   iii. Assuming a plant specific Simulator would be the input, Garbage in = garbage out. The alarm screen would need to be clear of useless or nuisance alarms in order to properly train ML.

   iv. If all of this is worked out and tested by Operators to show it works and is useful, I think the Regulator and Operators will trust it.

   v. There are some things to consider when implementing ML. I suggest you take the data stream from the field and separate it into two paths, one for the normal display and the other to the ML computer and a separate display. There would be a separate monitor that displays only the data from the ML computer and not the other data. Leave the normal display screen data alone and let it display all the data as it normally would. The Advantages:

      1. All the ML important data would fit on fewer screens and not be mixed in with other data. This would be a reference screen for the Operators to look at to get a better understanding of what is going on which meets the objective. It also leaves the normal display of information alone.

      2. It allows for isolation from the normal data the Operators receive. On the outside chance the field data is considered safety related (I doubt it though) the isolator to the ML computer can allow it to be isolated and non-safety related.

      3. The regulator may view this as additional information available to the Operators that doesn't interfere in the approved design and data flow path of information. My gut feeling from the little I've learned about ML is it is a great technology and would be accepted if properly developed and implemented.

Overall, while the primary goal of the SME's review was to determine whether Halden's approach was viable, the comments from the SMEs indicate that there is not enough information in the report to draw a definitive conclusion, and that there were a few issues with the work performed thus far that Halden can hopefully address moving forward. The explicit suggestions offered by the SMEs on how this research could be improved reinforces these first 2 themes.

Future research in this area will need to further validate the ML approach by comparing the state-based alarm solution to a state-based alarm system developed by human subject matter experts. This comparison would be a straightforward approach to performing the necessary step of further validating the effectiveness of using machine learning to develop a state-based alarm system. Differences in performance should be identified and provided as feedback to the developers of the ML approach to modify and improve the effectiveness of this technique.