

Light Water Reactor Sustainability Program

Bridging Equipment Reliability Data and Robust Decisions in a Plant Operation Context



June 2022

U.S. Department of Energy

Office of Nuclear Energy

DISCLAIMER

This information was prepared as an account of work sponsored by an agency of the U.S. Government. Neither the U.S. Government nor any agency thereof, nor any of their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness, of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. References herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the U.S. Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the U.S. Government or any agency thereof.

Bridging Equipment Reliability Data and Robust Decisions in a Plant Operation Context

D. Mandelli, C. Wang, R. Christian, E. Birch, S. Lawrence

June 2022

**Prepared for the
U.S. Department of Energy
Office of Nuclear Energy**

SUMMARY

In order to reduce operation and maintenance costs, nuclear power plants are moving from corrective and periodic maintenance to predictive maintenance strategies. Such a transition requires changes in the data that needs to be retrieved and the type of decision processes to be employed. Advanced monitoring and data analysis technologies are essential for supporting predictive strategies. They can in fact provide precise information about health of a component, track its degradation trends, and provide an estimate of its expected failure time. With such information, maintenance operations for a component can be performed right before its expected failure time. This dynamic context of operation and maintenance operations requires new methods to analyze data, propagate component health information from the component to the system level, and optimize plant resources. In this respect, the risk-informed asset management project has been tasked with developing and testing this new class of methods for a risk analytics toolset. This toolset consists of data analytics tools coupled with reliability methods designed to manage plant assets and performances in a predictive maintenance context. This report shows the latest improvements on this development and the initial testing of our methods on the three main research areas that the risk-informed asset management project is focusing on: equipment reliability data analytics, system reliability modeling, and plant resource optimization methods. We show how the methods developed in these areas can support predictive maintenance strategies by identifying the most critical components and setting an optimal maintenance schedule based on plant economic and operational constraints.

CONTENTS

SUMMARY	5
ACRONYMS	13
1. INTRODUCTION	15
1.1 Report Structure	15
2. A MODEL-BASED APPROACH TO PLANT HEALTH MANAGEMENT	16
2.1 Model-Based System Engineering Modeling	17
2.2 Causal Reasoning with OPM Models	18
2.2.1 Precursor Analysis	19
2.2.2 Cause-Effect Analysis	21
3. ANALYSIS OF ER DATA	21
3.1 NLP Analysis of Cause-Effect Paragraphs	22
3.1.1 Cause-Effect Paragraphs—Compound Sentence	25
3.1.2 Cause-Effect Paragraphs—Multiple Sentences	26
3.2 NLP Analysis of Health Status Reports	27
3.2.1 Analysis of Qualitative Observations	27
3.2.2 Analysis of Quantitative Observations	28
3.2.3 Analysis of Conjecture Observations	29
3.2.4 Identification of Temporal Attributes	30
3.2.5 Identification of Location Attributes	30
3.3 NLP Analysis of Temporal Relation Between Events	31
3.4 Examples of NLP Analysis	34
3.4.1 Rule-Based Health Information Extraction	34
3.4.2 Rule-Based Causal Relation Identification	36
3.4.3 Coreference Handling	37
3.5 NLP Methods Development	37
3.5.1 Tokenization	38
3.5.2 Sentence Segmentation	39
3.5.3 Part of Speech	39
3.5.4 Dependency Parsing	41
3.5.5 Lemmatization	43
3.5.6 Similarity	43
3.5.7 Rule-Based Entity Recognition	43
3.5.8 Merge Phrase	43
3.5.9 Coreference	43
4. RELIABILITY MODELING	44
4.1 Summary of Margin-Based Reliability Modeling	44
4.2 Notes on Distance Metrics	45
4.3 Margin Operations for AND, OR, KooN, and StandBy Operators	49
4.4 Integration of ER Data	50
4.4.1 Anomaly Detection Data	50
4.4.2 Condition-Based Data	51

4.4.3	Prognostic Data	56
4.5	Notes on Common Cause Failures	56
4.6	Examples of Margin-Based Reliability Calculations	57
4.7	Links Between Margin and Classical Reliability Approaches	60
4.8	Numerical Comparison Between Margin and Classical Reliability Approaches	61
5.	DECISION-MAKING	64
5.1	From Margin to Plant Resource Optimization Methods	64
5.2	Summary of Optimization Methods Development	65
5.3	Testing and Development of Evolutionary Methods	67
6.	CONCLUSIONS	70
	REFERENCES	71
	APPENDIX A: CLASSICAL RELIABILITY MODELING	74
	APPENDIX B: CUT SETS AND PATH SETS	76
	APPENDIX C: OPM MODELING	77
	APPENDIX D: MFW OPM MODELS	79
	APPENDIX E: MFW SYSTEM RELIABILITY DATA	83

LIST OF FIGURES

Figure 1.	Graphical overview of the RIAM project	16
Figure 2.	Graphical representation of a generic OPM model.	17
Figure 3.	Simplified representation of a centrifugal pump using OPM	18
Figure 4.	Pump OPM model and available monitoring data	19
Figure 5.	Causal diagram between monitoring data for the component shown in Figure 4.	19
Figure 6.	Given anomalous behaviors detected by the three sensors in the yellow area and provided the causal relationship among them (see Figure 5), the precursor can be identified as sensor Temperature_1	20
Figure 7.	Time of anomaly detection for the three sensors shown in Figure 6	20
Figure 8.	Given the identification of Temperature_1 as the sensor closest to the precursor, the candidate causes can be identified from the form elements of the component OPM model (see Figure 4) that support such function (highlighted in red).	21
Figure 9.	Graphical representation of elemental cause-effect structures: direct cause-effect association (top left), invalid association (top right), multiple causes and single effect association (center left), multiple effects and single cause association (center right), and causal homeostasis	24
Figure 10.	Example of compound sentence containing a causal relationship between two clauses	25
Figure 11.	Example of text containing a causal relationship between two sentences	27

Figure 12. Graphical concepts of time-based relations: order, duration, concurrence, and coincidence of events (https://archive.siam.org/meetings/sdm11/moerchen.pdf).....	32
Figure 13. SR ² ML NLP process.....	34
Figure 14. Tokenization process (source: https://spacy.io/usage/spacy-101).....	39
Figure 15. POS tagging and dependency parsing.	41
Figure 16. Graphical representation of event occurrences based on a margin framework.	46
Figure 17. Graphical representation of the margin calculation for $M(A \text{ AND } B)$ when considering the temporal evolution of MA and MB	47
Figure 18. Plot of the historic evolution of MA and MB and their predicted evolution based on the derivative information.	48
Figure 19. Plot of the historical and predicted evolution of MA (red line), MB (blue line), and $M(A \text{ AND } B)$ (green and purple line) based on the derivative information.....	48
Figure 20. Margin evolution for two components, A and B, in a standby configuration.	49
Figure 21. Margin calculation for KooN configuration.	50
Figure 22. Integration of anomaly detection data into margin calculation (AAKR test case).....	52
Figure 23. Typical current signature analysis where sideband currents I_{fsb} are centered around the frequency of the supply current. Source: https://irispower.com/learning-centre/relative-merits-off-line-line-testing-rotating-machine-stator-rotor-windings/	53
Figure 24. Graphical representation of margin provided measured sideband currents I_{fsb}	53
Figure 25. Time domain analysis of vibration signal. Source: Luo et al. (2021).	54
Figure 26. Vibration data for different mass flow rates and margin representation provided actual pump conditions. Adapted from Luo et al. (2021).	54
Figure 27. Example of scenario where normal and failed conditions (defined over a 2D space) are available to determine component margin.....	55
Figure 28. Vibration data recorded at the DE and FE under normal conditions and when two faults have been inserted (FD set to 0.007 in. and 0.040 in.). Source: https://engineering.case.edu/bearingdatacenter	56
Figure 29. Margin values obtained from the two proposed approaches (green and blue lines) given an estimate of component's RUL (red line).	57
Figure 30. Graphical representation of common causes in a margin context.	58
Figure 31. Example of system architecture represented in terms of block diagrams (Youngblood 2001).	58
Figure 32. Example of margin-based calculations using prognostic data for the system indicated in Figure 31. For each of the seven components, A–G, a RUL is provided in the top plot and the corresponding system margin quantification is indicated in the bottom plot.	59
Figure 33. Plot of RIM measures for the seven components of the system of Figure 31 given the provided prognostic data (in terms of RUL) indicated in the top plot of Figure 32.....	60
Figure 34. Comparison between margin-based and probability of failure-based reliability modeling approaches.	61
Figure 35. Evolution of M_{sys} for two components in a series configuration.	63

Figure 36. Evolution of <i>Msys</i> for two components in a parallel configuration.	63
Figure 37. Evolution of <i>Msys</i> for two components in a standby configuration.	64
Figure 38. Graphical representation of urgency given estimated component failure and required restoration time.	65
Figure 39. Selection of the most critical components in an urgency-importance diagram.	65
Figure 40. Classes of maintenance approaches and their corresponding optimization methods.	67
Figure 41. Graphical representation of the main GA workflow.	68
Figure 42. Plot representing the temporal evolution of the population input (i.e., <i>a, b, c, d</i>) and output (i.e., <i>ans</i>) variables.	69
Figure 43. Parallel plot representing population values in the input space (represented by four variables ranging from a to d) for different GA iterations (indicated with batch numbers).	70
Figure 44. Graphical representation of the most common block configurations: series (top left), parallel (top right), standby (bottom left), and KooN (bottom right).	75
Figure 45. Schematics of a PWR MFW system (source: www.nuclear-power.com).	79
Figure 46. OPM model of MFW system.	80
Figure 47. OPM model of a turbine-driven pump.	80
Figure 48. OPM model of a motor-operated valve.	81
Figure 49. OPM model of a shaft bearings.	81
Figure 50. OPM model of a feedwater heater.	82
Figure 51. OPM model of a deaerator.	82

LIST OF TABLES

Table 1. Component ER data: types, structures, and examples.	22
Table 2. List of paragraph types (Swales and Feak 2012).	23
Table 3. Partial list of keywords that indicate a cause-effect paragraphs.	24
Table 4. List of structures that indicate a cause-effect paragraphs.	25
Table 5. List of relations that indicate a cause-effect paragraphs.	25
Table 6. NLP steps to extract a causal relationship between clauses in a single sentence.	26
Table 7. List of transition keywords that indicate a causal relationship between clauses in a single sentence.	26
Table 8. List of sentence relations for qualitative observation.	27
Table 9. Partial list of keywords that indicate negative information.	28
Table 10. Partial list of keywords that indicate positive information.	28
Table 11. Partial list of keywords that indicate neutral information.	28
Table 12. List of sentence relation for quantitative observation.	29

Table 13. List of keywords that indicate a conjecture observation.....	29
Table 14. List of relations that indicate a conjecture observation.	30
Table 15. List of approximations that indicate a temporal attribute.	30
Table 16. List of relations that indicate a temporal attribute.	31
Table 17. List of keywords that indicate a location attribute.	31
Table 18. List of relations that indicate a location attribute.....	32
Table 19. List of keywords and structures that indicate order of events.	33
Table 20. List of keywords that indicate the concurrence and coincidence of events.....	33
Table 21. List of relations that indicate the order of events.....	33
Table 22. List of relations that indicate the concurrence and coincidence of events.....	33
Table 23. Examples for IR textual reports.	34
Table 24. OPM elements identified in the IRs of Table 23 from the centrifugal pump OPM model shown in Figure 4.	35
Table 25. Information extraction by SR ² ML NLP module.....	35
Table 26. Extracted SSC entities and their health status.....	36
Table 27. Causal relations identified by SR ² ML NLP module.	37
Table 28. Example of coreference identification.	37
Table 29. NLP analysis pipelines.....	38
Table 30. Part-of-speech tags (source: https://v2.spacy.io/api/annotation).	40
Table 31. Syntactic dependency parsing (source: https://v2.spacy.io/api/annotation).	42
Table 32. Comparison between system failure time T_{sys} and system margin M_{sys} for four system configurations.	62
Table 33. Summary of classical reliability calculations for the considered four configurations.....	75
Table 34. Lists of MCSs and MPS for the system shown in Figure 31.	76
Table 35. Basic elements of OPM (adapted from Dori and Crawley [2002]).	77
Table 36. Causal relations from OPM diagrams.....	78

ACRONYMS

AAKR	auto-associative kernel regression
BE	basic event
ER	equipment reliability
ET	event tree
FT	fault tree
GA	genetic algorithm
INL	Idaho National Laboratory
IR	issue report
KNN	k nearest neighbor
KooN	K out of N
LWR	light-water reactor
MBSE	model based system engineering
MCS	minimal cut set
ML	machine learning
MPS	minimal path set
NLP	natural language processing
NPP	nuclear power plant
OPM	object-process methodology
POS	part of speech
PWR	pressurized-water reactor
RIAM	risk-informed asset management
RUL	remaining useful life
SysML	Systems Modeling Language
SR ² ML	Safety, Risk, Reliability Model Library
SSCs	structures, systems, and components

Bridging Equipment Reliability Data and Robust Decisions in a Plant Operation Context

1. INTRODUCTION

To reduce operation and maintenance costs, existing nuclear power plants (NPPs) are moving from corrective and periodic maintenance to predictive maintenance strategies. While corrective maintenance is performed only when the component fails (with high costs due to component replacement and unexpected system and plant unavailability, e.g., loss of generation), periodic maintenance is performed at specific time intervals based on reliability factors and past operational experience (with high costs due to continuous maintenance operations that may not be warranted).

The transition from periodic or corrective maintenance to predictive performance is designed so that maintenance occurs only when the component requires it (e.g., before its imminent failure). This guarantees that component availability is maximized and that maintenance costs are minimized. However, these benefits require changes in the data that needs to be retrieved and the type of decision processes to be employed. Advanced monitoring and data analysis technologies are essential to support predictive strategies. They can in fact provide precise information about health of a component, track its degradation trends, and provide information of its expected failure time. With such information, maintenance operations for a component can be performed right before its expected failure time. This dynamic context of maintenance operations (i.e., predictive) requires new methods to analyze data, propagate component health information from the component to the system level, and optimize plant resources.

The risk-informed asset management (RIAM) project has been tasked with developing and testing this new class of methods for a risk analytics toolset. This toolset consists of data analytics tools coupled with reliability methods designed to manage plant assets and performances in a predictive maintenance context. This report shows the latest improvements on this development and initial testing of our methods on the three main research areas that the RIAM project is focusing on, see Figure 1: equipment reliability (ER) data analytics, system reliability modeling, and plant resources optimization methods. We show how the methods developed in these areas can support optimization of predictive maintenance strategies by identifying the most critical components and setting an optimal maintenance schedule based on plant economic and operational constraints.

1.1 Report Structure

This report has been structured in three main parts that cover the three areas of research and development under the RIAM project. Section 2 presents our model-based approach to analyze ER data and its foundational elements. We provide details on how systems engineering models can be used to analyze ER data more effectively and extract information from textual data. Section 3 focuses on the development and testing of ER data analyses designed extract quantitative information to automatically assess component health, identify anomalies, and assess possible causes for such anomalies. Section 4 expands reliability modeling methods developed in the past years based on the concept of margins. We provide some practical examples on how available ER data can be used to assess component margins and a detailed description on how a margin-based reliability approach contrasts a classical probability-based reliability approach. Lastly, Section 5 provides details on how the results obtained from the margin-based approach can identify the most critical component that require maintenance operations. In addition, we describe progress in the optimization methods designed to manage plant resources. During Fiscal Year (FY) 2022, we focused, in particular, on the development and testing of evolutionary methods (i.e., based on

genetic algorithms [GAs]) that are part of the Risk Analysis and Virtual ENvironment (RAVEN) software platform (Alfonsi et al. 2020). Moreover, we developed few plotting tools designed to assess the performance of the employed optimization method and its convergence status.

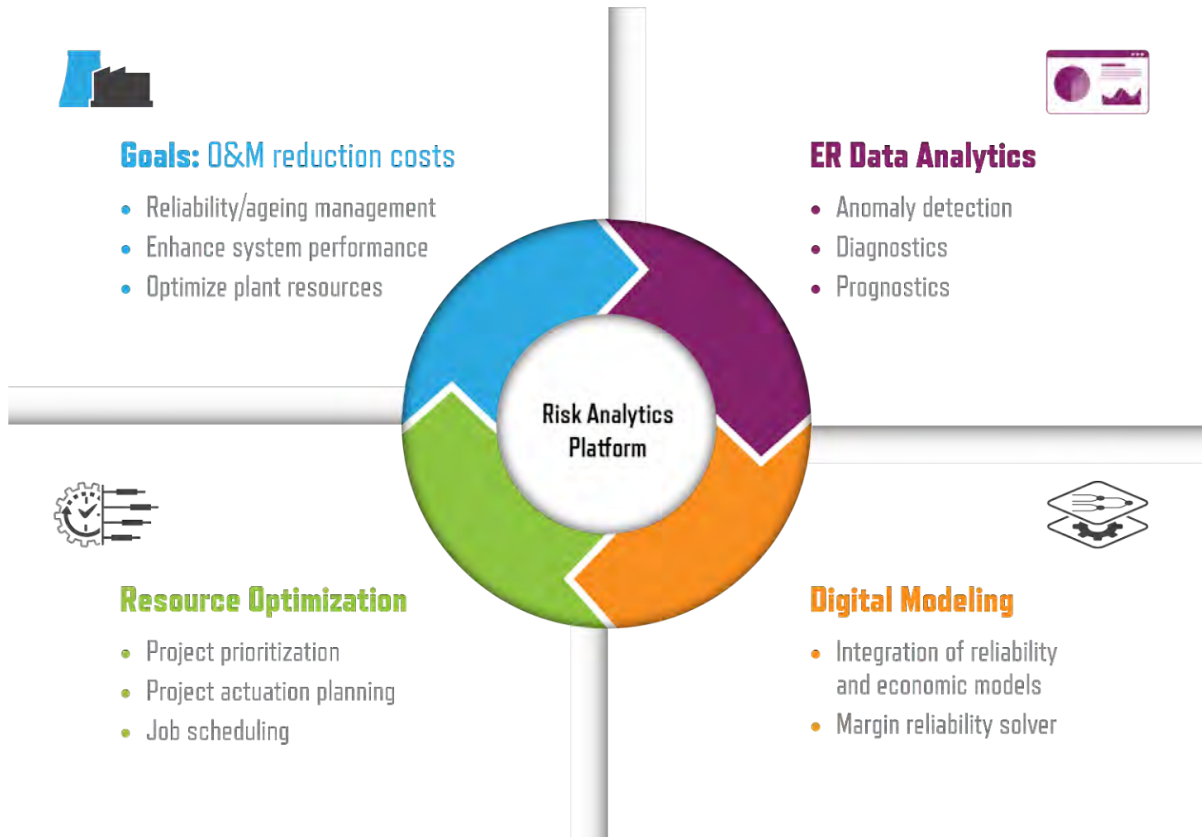


Figure 1. Graphical overview of the RIAM project.

2. A MODEL-BASED APPROACH TO PLANT HEALTH MANAGEMENT

As indicated in detail by Mandelli et al. (2021), ER data can have heterogenous data formats: textual, numeric, image, etc. Plant system engineers possess vast knowledge of the system, its component dependencies and architecture, and system’s past and current performance. As such, they have many methods to interpret ER data, identify the causal links between events, and plan recovery actions. Is it possible for a machine to assist a system engineer in performing these tasks more efficiently or more accurately? While many computer-assisted methods are available they may not be entirely suitable for the task in hand. Currently, data analysis methods are based on machine learning (ML) techniques (Mohri and Rostamizadeh 2012) that focus on finding patterns contained in data. While such approaches are valuable for some use cases, as shown in (Nassif et al. 2021), not all patterns provide insights about the system. This is often translated as: “correlation does not imply causation.”

Here, we are looking at computational methods designed to support system engineers with the analysis of ER data using machine reasoning (rather than ML) methods. Machine reasoning is based on the construction of causal relations between data elements. This construction process cannot rely solely on data, as it requires models. These models represent specific aspects of the “real world” and are the foundations

to perform such causal reasoning. When dealing with ER data, these models need to emulate system engineer knowledge about component and system architecture and dependencies.

2.1 Model-Based System Engineering Modeling

In this project we rely on model-based system engineering (Borky and Bradley 2018) (MBSE) representations of systems and components. This representation is based on diagrams designed to set dependencies (through links) between the “form” and “function” elements of the considered system and component. While several MBSE languages are available to construct such diagrams, e.g., System Modeling Language (SysML) (Friedenthal et al. 2008), we chose the Object-Process Methodology (OPM) language (Dori and Crawley 2002). An OPM diagram provides an essential description of a component from both a form and functional perspectives. Each diagram (refer to the generic diagram shown in Figure 2) explicitly indicates how the component internal functions (represented as ellipses) act upon operands and how the elemental parts (represented by square boxes) support these functions.

From an ER perspective, monitoring and testing activities generate information about component functions (e.g., rpm recorded for an induction motor) and form (e.g., blade corrosion of the centrifugal pump) elements. Aging and degradation processes directly alter the form-related elements of the component that consequently affect component functional elements (i.e., the functions of Figure 2). Typically, from a reliability perspective, component failure modes are described in term of loss of function, and, hence, in the OPM diagram, failure modes are only directly linked to the functional elements of the component. Lastly, note that maintenance activities (such as component replacement, refurbishment, or reconditioning) act on the form elements of components.

We chose the OPM modeling language based on the following reasons:

- The OPM language is relatively simple in nature and provides the most basic functionalities required to extract causal relations between data elements
- OPM diagrams are easy to process digitally and to use to generate graph structures
- A direct link between the OPM model and ER data (of any form, e.g., textual or numeric) along with aging and degradation can be uniquely established.

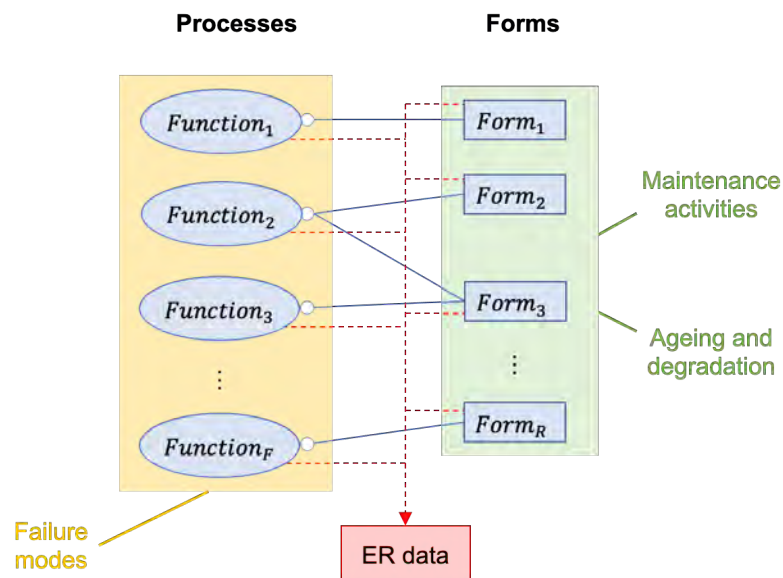


Figure 2. Graphical representation of a generic OPM model.

For the scope of this report, the OPM diagram of a component (or a system) represents the key point to automatically understand and analyze health data. In particular, it clearly links monitored and recorded data with failure modes that might affect component performance and maintenance activities that would restore component functionality. We are employing model-based data analysis methods to link component models with data rather than using ML methods, which solely rely on available data to perform diagnostic and prognostic operations. Note that an OPM diagram extends failure modes and effect analysis tables by providing a form and functional description of the considered system in a graphical form.

An example of an OPM model for a centrifugal pump is provided in Figure 3. This simple representation includes the most basic elements found in most OPM models and provides the following information:

- The form element “centrifugal pump” is composed by (through the composition link) four elements: shaft, impeller, bearing, and motor.
- The function “increase fluid pressure” requires the form element “centrifugal pump” (through the instrument link).
- The function “increase fluid pressure” transforms “fluid pressure” from low to high (through the transformation link).
- “Fluid pressure” is an attribute of the form element “fluid” (through the characterization link) that pump is affecting.

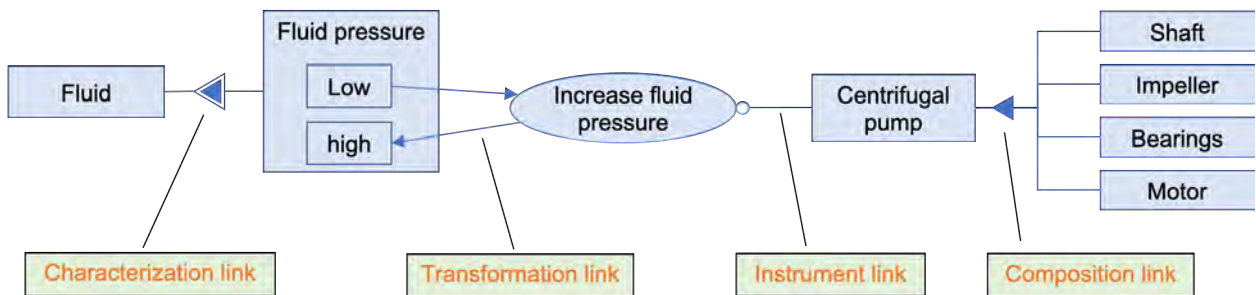


Figure 3. Simplified representation of a centrifugal pump using OPM.

2.2 Causal Reasoning with OPM Models

This section shows how causal reasoning can be performed using OPM models and available monitoring data. For this purpose, we considered the OPM model of a generic centrifugal pump shown in Figure 4, which also include the location of specific monitoring data. Given the structure of the OPM model and based on the links defined in it, it is now possible to automatically create the graph shown in Figure 5. Such a graph explicitly indicates the causal relationship between the monitored physical variables. Appendix C provides more detail regarding the construction of a causal relationship between OPM elements.

Given the diagrams shown in Figure 4 and Figure 5, it is possible to perform the following analyses:

- *Precursor analysis*: given a set of events (e.g., abnormal behavior), this analysis will identify which event triggered all the others (i.e., the precursor), see Section 2.2.1.
- *Cause-effect analysis*: given a recorded event (e.g., a single abnormal event or precursor event identified above), this analysis will identify the possible cause(s) of such an event and inform on current component health conditions, see Section 2.2.2. In an OPM context, the cause(s) will focus

on form elements (see also Figure 2). Note that when combined with the previous analysis, this process can also inform on the evolution of future occurrences.

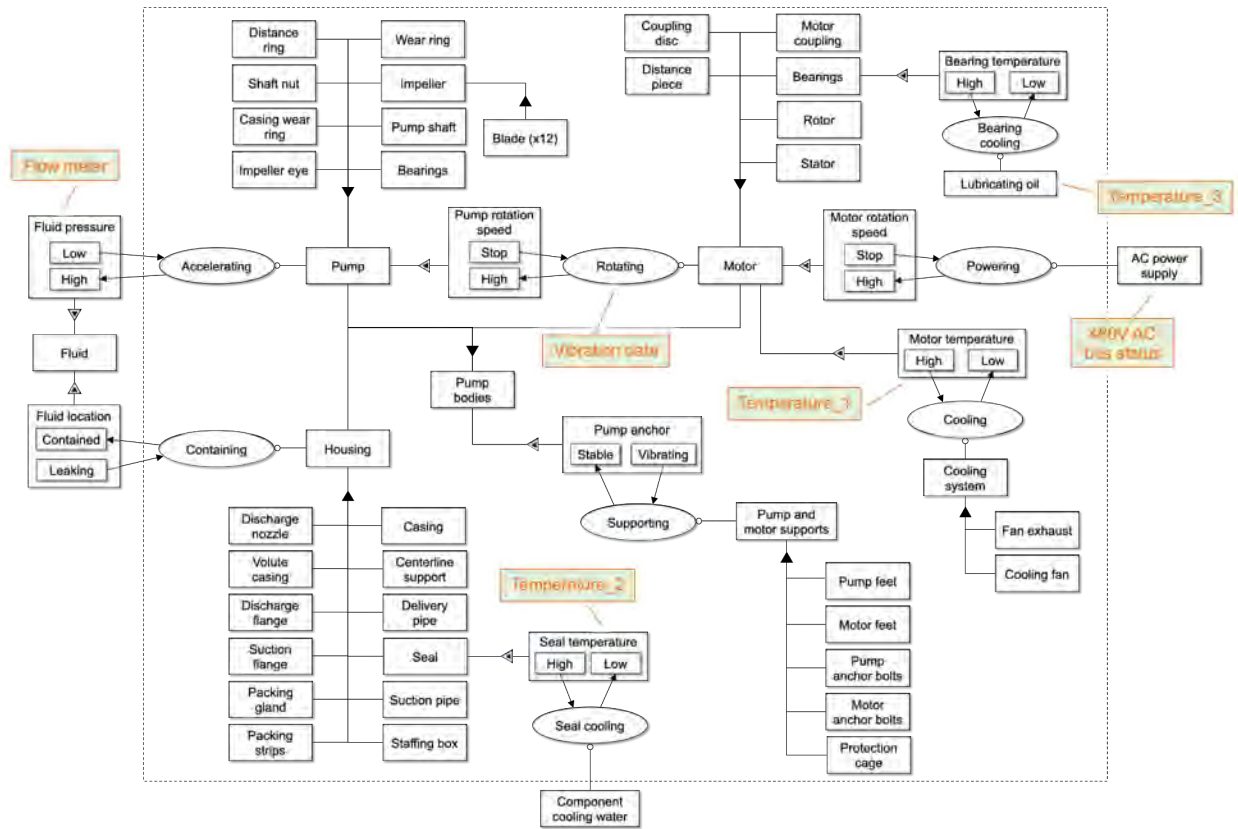


Figure 4. Pump OPM model and available monitoring data.

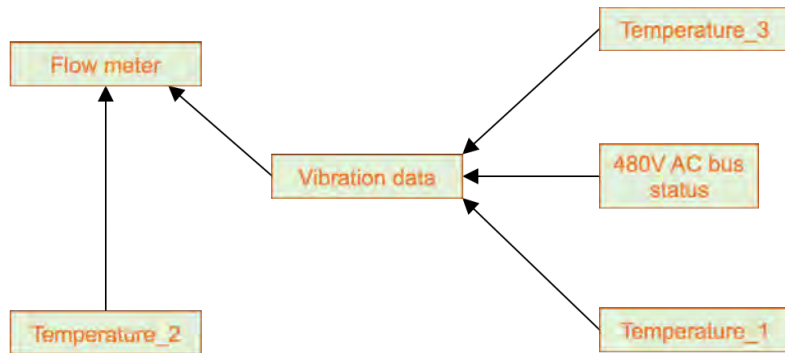


Figure 5. Causal diagram between monitoring data for the component shown in Figure 4.

2.2.1 Precursor Analysis

Given the abnormal behavior of multiple monitoring data sources, the precursor is determined by looking at the source of the causal path between these data sources. The required information for this analysis is contained in the sensors causal diagram (e.g., Figure 5). As an example, if abnormal behaviors

are being observed from the Flow meter, Vibration data, and Temperature_1 sensors, the precursor is identified as Temperature_1 since it is the source of the causal path between these three sensors, see Figure 6; in other terms:

$$\text{Temperature}_1 \rightarrow \text{Vibration data} \rightarrow \text{Flow meter} \quad (1)$$

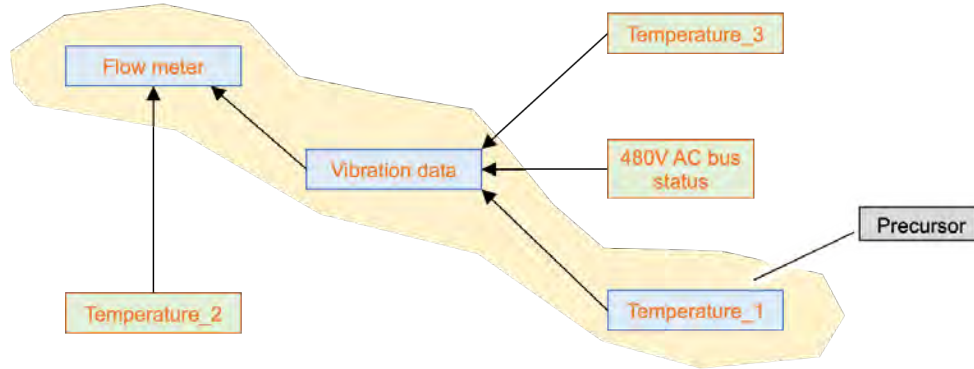


Figure 6. Given anomalous behaviors detected by the three sensors in the yellow area and provided the causal relationship among them (see Figure 5), the precursor can be identified as sensor Temperature_1.

Note that abnormal behaviors can be identified by anomaly detection methods applied to numeric data (Nassif et al. 2021) and textual data (see Section 3). Hence, the ability to correctly process both numeric and textual data is essential to reconstruct the complete chain of events (assuming actual ER data is available).

In practical settings, it is not uncommon that an anomaly recorded by a sensor might have a countereffect to other elements of the same system and component that does not have a direct causal relationship. An example is provided in Figure 7 that indicates the time of anomaly detection for the same three sensors shown in Figure 6. This timing information is directly employed to refine the causal reasoning since cause precedes the effect on a temporal scale. In this scenario, we can obtain the following causal relations:

$$\begin{aligned} \text{Vibration} &\rightarrow \text{Temperature}_1 \\ \text{Vibration} &\rightarrow \text{Water_flow} \end{aligned} \quad (2)$$

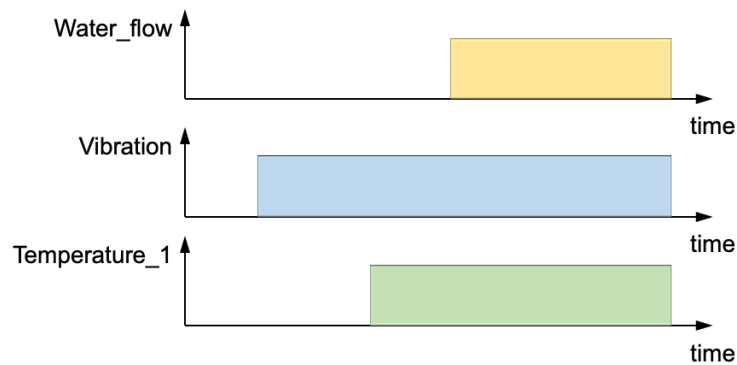


Figure 7. Time of anomaly detection for the three sensors shown in Figure 6.

2.2.2 Cause-Effect Analysis

Given the abnormal behavior of one monitoring data source (e.g., the precursor identified from the precursor analysis shown above), the set of candidate causes is identified by looking at the form elements that support the monitored function. As an example, if Temperature_1 is reporting anomalous behavior and Temperature_1 has also been identified as the precursor (see precursor analysis presented above), the possible cause for such an anomaly can be identified in the cooling system (see Figure 8) in either the fan exhaust or cooling fan.

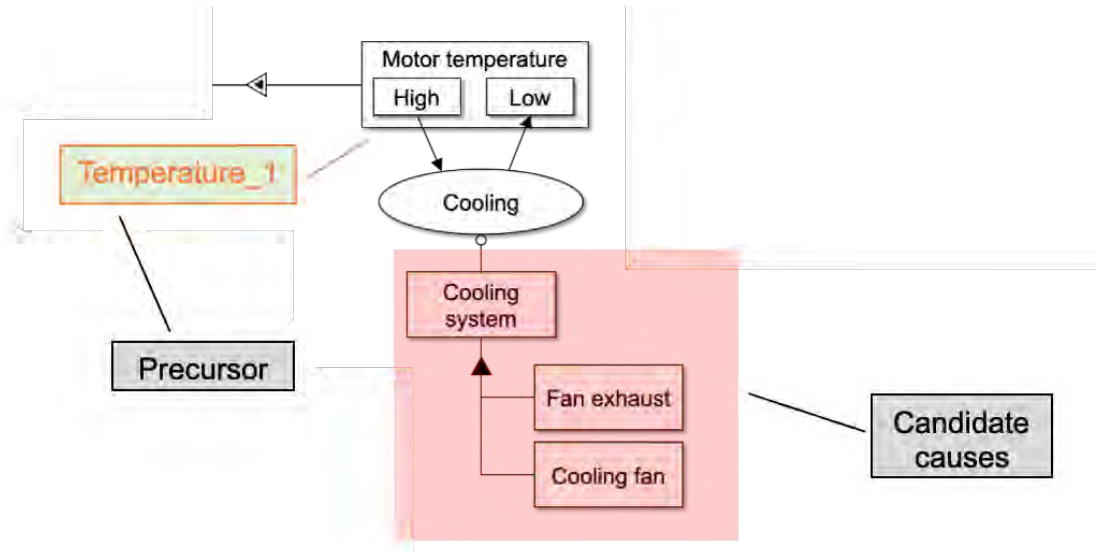


Figure 8. Given the identification of Temperature_1 as the sensor closest to the precursor, the candidate causes can be identified from the form elements of the component OPM model (see Figure 4) that support such function (highlighted in red).

3. ANALYSIS OF ER DATA

As indicated in Section 1, availability of ER data is an essential element to support decisions designed to maximize plant availability. In this respect, NPPs are constantly monitoring the status and performance of many systems and components. Hence, a large amount of data is continuously being generated. Table 1 provides a summary of the types of data available from existing power plants, and for each type, we provide the format (e.g., either numeric or textual) for such data and examples. The analysis of numeric ER data has been addressed in many works (Xingang et al. 2021) and applied to many operational directions including anomaly detection, diagnosis, and prognosis. The analysis of textual data has been investigated only recently using ML methods (Young et al. 2018) designed to assess their nature (e.g., safety or non-safety related).

In the RIAM project, we aim to solve a different class of problems that requires reasoning rather than data learning. We are in fact addressing the analysis of ER data by focusing on:

- Causal reasoning from numeric and textual data (see Section 3.1)
- Knowledge extraction from textual data (see Section 3.2)

The objective of analyzing textual ER data is to automate the extraction of quantitative knowledge from textual data and assist system engineers in assessing system health. The concept of “knowledge extraction” is very broad, and its definition might vary depending on the application context. From a linguist point of view, the development of a paragraph (or a sentence) as part of the text follows several directions, as

indicated in Table 2 (Swales and Feak 2012). Table 2 also provides the relevance of each paragraph to an actual plant reliability context based on our initial observations from actual plant ER textual data. The list indicated in Table 2 is incomplete due to the complex nature and possible structure ramifications of any natural language (e.g., English).

Table 1. Component ER data: types, structures, and examples.

Type	Data Structure	Examples
Health data	Textual Numeric Images (in some instances)	Report of component failure Shaft vibration data Report of heavy corrosion of impeller
Performance data	Textual Numeric	Successful startup of component and verification of water flow pressure Pump power vs. flow curve
Boundary conditions	Textual Numeric	Fluid found nearby component Environment temperature

Given the observations listed in Table 2, we have focused our attention to three classes of paragraphs that are common from ER textual data:

- Causal relation between events (refer to cause-effect paragraph), see Section 3.1
- Health status report on a single event (refer to analysis and problem-solution paragraphs), see Section 3.2
- Time-based relationship between events (refer to narration paragraphs), see Section 3.3.

Before jumping on the actual developed NLP methods, from a linguistic point of view, it is important to identify specific terms we use throughout this report:

- *Text*: the actual raw content of an incident record (IR) that can be composed of multiple sentences
- *Sentence*: the base element of a text that expresses a complete concept, which can be simple (i.e., single clause) or complex (i.e., multiple clauses)
- *Clause*: a grammatical constituent that consists of a subject and a predicate.

As mentioned, the goal of our NLP methods is to extract quantitative knowledge from the three classes of paragraph listed above. Hence, ML methods based on a supervised or unsupervised algorithm do not really suit our scopes since they only provide qualitative information (e.g., which user-specified class a sentence belongs to).

Our approach relies in a small portion on ML methods and is predominantly rule based. More specifically, for each of the three classes of paragraphs listed above, our NLP methods are looking within each sentence and paragraph at specific keywords, sentence architecture relations, and structures.

3.1 NLP Analysis of Cause-Effect Paragraphs

A common pattern in textual ER data is a report of multiple events along with a causal relationship among them. In its simplest form, this paragraph contains an event (i.e., the cause) that triggered a second event (i.e., the effect). However, the structure of this type of paragraph can have different forms. In this respect, Figure 9 presents, in graphical form, the main forms of a cause-effect structure:

- An event that has been identified as not the cause of another event (i.e., an invalid causal association)
- Multiple causes that trigger a single effect
- A single cause that triggers multiple effects
- A causal homeostasis that identifies a chain of events that perpetuates in a continuous loop.

Table 2. List of paragraph types (Swales and Feak 2012).

Paragraph Type	Description	Relevance to ER Textual Data
Exemplification	Designed to provide clarification about a topic	Low
Narration	Designed to indicate a series of events that have occurred	High
Process	Designed to develop sequences that describe how an activity can be performed	Medium
Description	Designed to provide details about the subject	Low
Comparison and contrast	Designed to examine similarities and dissimilarities between two subjects	Low
Analogy	Designed to explain the subject in terms of another	Low
Cause-effect	Designed to provide a causal relationship between two entities	High
Classification and division	Designed to describe entities that are part of a whole	Low
Definition	Designed to set the definition boundaries of the subject	Low
Analysis	Designed to describe the subject by weighting evidence and possible causal links	High
Enumeration	Designed to itemize a series of objects	Low
Problem-solution	Designed to describe an undesired situation and provide a way to restore it	High

Our approach is not employing classical NLP methods (Young et al. 2018) based on ML algorithms (e.g., through classification methods [Mohri and Rostamizadeh 2012]). Our methods are rule based (Doan et al. 2019) since our goal is to extract actual quantitative information from textual data rather than “classifying” the nature of the raw text. These rules are based on the identification of:

- Keywords, such as nouns, verbs, and adverbs, that identify the possibility that the sentence might contain a causal relation between the subject(s) and the object(s) contained in that sentence (see Table 3)
- NLP structures (or constructs) composed of multiple words that indicate a casual transition between clauses contained in a sentence or between sentences (see Table 4)
- Relations between sentence subjects and verbs that are designed to reconstruct the node (see Table 5): cause → effect.

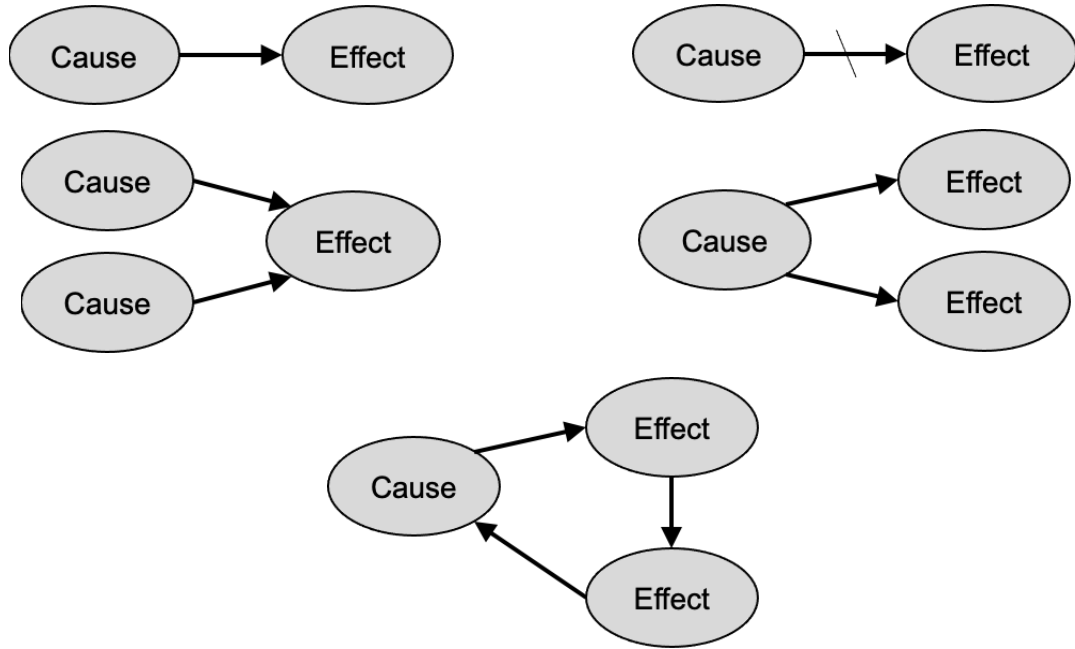


Figure 9. Graphical representation of elemental cause-effect structures: direct cause-effect association (top left), invalid association (top right), multiple causes and single effect association (center left), multiple effects and single cause association (center right), and causal homeostasis.

Table 3. Partial list of keywords that indicate a cause-effect paragraphs.

Nouns	Verbs	Adverbs
Aftereffect	Augment	Afterwards
Aftermath	Backfire	Consequently
Backfire	Begin	Eventually
Byproduct	Bring about	Finally
Conclusion	Build-up	Hence
Consequence	Cause	So
Counteraction	Change	Subsequently
Counterbalance	Combat	Then
Countermove	Compensate	Therefore
Effect	Counter	Thus
	Create	Ultimately
	Deactivate	
	Decelerate	
	Decrease	

Table 4. List of structures that indicate a cause-effect paragraphs.

Structures
In response to
Attributed to
As a result of
For this reason
In consequence
In this way
In such a way

Table 5. List of relations that indicate a cause-effect paragraphs.

Relations	DAG
Event_A + “causal verb” (active) + Event_B	$A \rightarrow B$
Event_A + “causal verb” (passive) + Event_B	$B \rightarrow A$
Event_A + [to be] a “causal noun” + Event_B	$A \rightarrow B$
Event_A + [to be] a “effect noun” + Event_B	$B \rightarrow A$
The “causal noun” of + Event_A + [to be] + Event_B	$B \rightarrow A$
The “effect noun” of + Event_A + [to be] + Event_B	$A \rightarrow B$
Clause_A ; + “cause/effect structure” + Clause_B	$A \rightarrow B$ or $B \rightarrow A$
“Cause/effect structure” + Clause_A ; + Clause_B	$A \rightarrow B$ or $B \rightarrow A$
Clause_A . “Cause/effect structure” + Clause_B	$A \rightarrow B$ or $B \rightarrow A$
Event_A + (verb, “causal adverb”) + Event_B	$A \rightarrow B$

3.1.1 Cause-Effect Paragraphs—Compound Sentence

Here we discuss a case where a single sentence contains multiple clauses (typically two or three) linked together by a causal relationship. An example of a compound sentence with a causal relationship between two clauses is provided in Figure 10. In that example, note that the causal structure “for that reason” is creating a causal relationship between two events contained in two separate clauses. Each clause is then processed using the NLP methods presented in Section 3.4. The NLP analysis workflow designed to extract the causal relationship between clauses contained in a single sentence is shown in Table 6. In addition, Table 6 provides the outcome of each step for the example indicated in Figure 10.

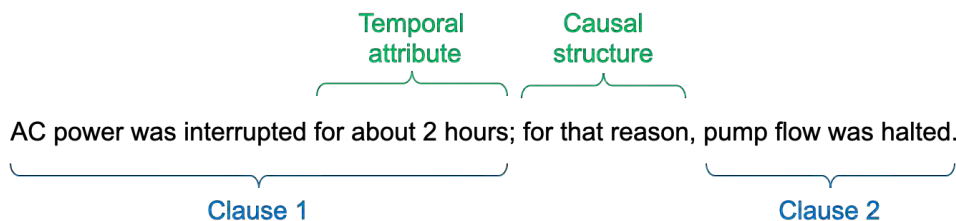


Figure 10. Example of compound sentence containing a causal relationship between two clauses.

Table 6. NLP steps to extract a causal relationship between clauses in a single sentence.

ID	Step	Example (see Figure 10)
1	Identify clauses from sentence	Clause 1 = AC power interruption Clause 2 = pump flow halted
2	Identify transition keywords (see Table 7)	“for that reason”
3	Process each clause (see Section 3.3)	Element_1 = (AC power, degraded) Element_2 = (pump flow, degraded)
4	Create corresponding directed acyclic graph	Element_1 → Element_2

Table 7. List of transition keywords that indicate a causal relationship between clauses in a single sentence.

Structures
Accordingly
Consequently
Hence
On account of
So
As a result
Due to
If ... then
Results in
Therefore
Since
Thus
Because of
For that reason
Leads to
As such
It follows that
Thereupon
Ergo
Being that
So that

3.1.2 Cause-Effect Paragraphs—Multiple Sentences

Here we discuss the case where multiple sentences (which might contain multiple clauses) linked together by a causal relationship. An example of text containing a causal relationship between two sentences is provided in Figure 11. From that example, note that the causal structure “consequently” is creating a causal relationship between two events in two separate sentences. Each sentence is then processed using the NLP methods presented in Section 3.2 and Section 3.3. The NLP analysis workflow designed to extract a causal relationship between multiple sentences is similar to the one shown in Table 6.

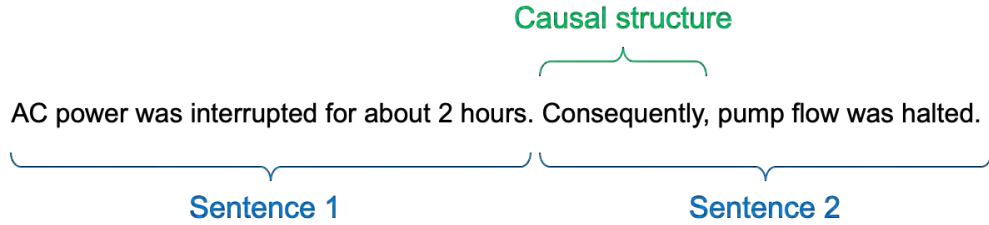


Figure 11. Example of text containing a causal relationship between two sentences.

3.2 NLP Analysis of Health Status Reports

The analysis of a health status report can have different forms. By looking at an initial textual data set, we were able to identify three types of health status reports:

- *Qualitative observation*: the report provides a qualitative observation (e.g., good, degraded, increase, decrease, stable, stop) about an event (see Section 3.2.1).
- *Quantitative observation*: the report provides a precise observation (i.e., report on point value or delta estimate of a measured variable) of an event (see Section 3.2.2).
- *Conjecture observation*: the report provides information about a future prediction or hypothesis about past (see Section 3.2.3).

In addition, we have identified two attributes that might be contained in health status reports that we aim to extract from the raw text: temporal (see Section 3.2.4) and location attributes (see Section 3.2.5).

3.2.1 Analysis of Qualitative Observations

A report in this category provides a fairly simple (i.e., qualitative) observation about an event. As indicated by Mandelli et al. (2021), the starting point is the creation of a full set of relations. Table 8 provides a list of relations that have been identified along with ER related examples. As indicated in Table 8, the set of relations are based on specific sets of nouns, adjectives, verbs, and adverbs. Given the nature of these observation, each of these grammatical entities (i.e., nouns, adjectives, verbs, and adverbs) can convey a qualitative information: positive, negative, or neutral. In this respect, Table 9, Table 10, and Table 11 provide a subset of grammatical entities for each of the three classes (positive, negative, or neutral).

Table 8. List of sentence relations for qualitative observation.

Relation	Example
Subj + “status verb”	Pump was not functioning
Subj + “status verb” + “status adjective”	Pump performances were acceptable
Subj + “status verb” + “status adverb” + obj	Pump was partially working
“status adjective” + subj + “status verb”	Unresponsive pump was observed
“status noun” + “prep” + “status verb”	Deterioration of pump impeller was observed

Table 9. Partial list of keywords that indicate negative information.

Nouns	Verbs	Adjectives	Adverbs
Breakdown	Disabled	Unacceptable	Inaccurately
Collapse	Reject	Improper	Erroneously
Decline	Stop	Inadmissible	Wrongly
Deficiency	Block	Undesirable	Inadequately
Deterioration	Halt	Unsatisfactory	Incompletely
Failing	Oppose	Unacceptable	Partially
Decay	Inhibit	Unsuitable	Imperfectly
Downfall	Hinder	Unwanted	

Table 10. Partial list of keywords that indicate positive information.

Nouns	Verbs	Adjectives	Adverbs
Accomplishment	Enable	Ready	Accurately
Achievement	Empower	Fit	Nicely
Enhancement	Facilitate	Capable	Perfectly
Progression	Permit	Apt	Precisely
Solution	Set up	Available	Properly
	Endow	Adequate	Rightly
	Let	Competent	Accurately
	Make	Proficient	Appropriately

Table 11. Partial list of keywords that indicate neutral information.

Nouns	Verbs	Adjectives
Analysis	Inspect	Acceptable
Assessment	Monitor	Usable
Diagnosis	Measure	Attainable
Evaluation	Witness	Consistent
Exploration	Examine	Constant
Investigation	Note	Stable
Probe	Recognize	Unaffected
	View	Uninterrupted
	Watch	Untouched
		Intact

3.2.2 Analysis of Quantitative Observations

This kind of reports provides a precise observation (i.e., a measured point value or delta estimate) of a measured variable. This observation requires a numeric value followed by its unit; however, it is not unusual that the unit might be missing. The structural relations for this kind of report are often simple in nature, and the identified relations are listed in Table 12. Note that when referring to delta estimates, verbs and nouns convey a qualitative information (positive, negative, or neutral) can be present.

Table 12. List of sentence relation for quantitative observation.

Relation
[neutral verb] + “quantity value”
[neutral verb] + “quantity delta value”
“quantity value” + [neutral noun]
“quantity delta value” + [neutral noun]
[negative verb] + “quantity value”
[negative verb] + “quantity delta value”
“quantity value” + [negative noun]
“quantity delta value” + [negative noun]
[positive verb] + “quantity value”
[positive verb] + “quantity delta value”
“quantity value” + [positive noun]
“quantity delta value” + [positive noun]

3.2.3 Analysis of Conjecture Observations

This kind of report provides information about future prediction (e.g., an event that can occur in the future) or hypothesis about past events (e.g., a failure that might have occurred). In this context, the verb tense plays a role in the identification of this kind of report. Future predictions are characterized by present and future tense verbs; hypotheses about past events instead are typically characterized by past tense verbs. Also, for this kind of reports, we have identified specific keywords (see Table 13) and relations (see Table 14) that can inform our methods that we are dealing with a conjecture observation.

Table 13. List of keywords that indicate a conjecture observation.

Keyword
Expected
Possible
Probable
Feasible
Plausible
Presumed
Hypothetical(ly)
Likely
Unlikely
Potential
Uncertain
Anticipated
Foreseen
Impending

Upcoming
Brewing
Looming
Forthcoming

Table 14. List of relations that indicate a conjecture observation.

Relation	Example
Subj + “future verb”	The pump will fail
Subj + “conjecture keyword” + “verb”	The pump is likely to fail
Conditional + subj + “verb” + “conjecture keyword” + “verb”	If the pump overheats, it is expected to fail
Subj + “past verb” + hypothesis	The pump failed because it overheated

3.2.4 Identification of Temporal Attributes

Temporal attributes indicate time instances when specific event have occurred. Time of occurrence is an important factor from a causal point of view since the emergence of an effect is always preceded by its cause. Hence, temporal information can be valuable to identify the causal links between recorded events. The identification of temporal attributes is being performed by looking at specific prepositions and relations that are listed in Table 15 and Table 16, respectively.

Table 15. List of approximations that indicate a temporal attribute.

Approximation
About
Almost
Nearly
Roughly
Approximately
Nearly
Around
Closely
Circa
Close
Like
More or less
Plus or minus
Roughly

3.2.5 Identification of Location Attributes

Location attributes provide qualitative information where specific events have occurred. While location information does not provide additional health information to a system engineer, it might contain clues about the health of a specific component when a reported event has occurred near it. As an example, the textual report

“An oil puddle was found nearby pump MFW-1A.”

identifies an element (i.e., oil) that is an integral part of the OPM diagram (see Figure 4) of the considered component (i.e., MFW-1A pump) located nearby such a pump. Thus, this textual report indicates how the oil element is no longer part of the OPM diagram. The same OPM diagram can now be used to infer the impact of such an event on pump function. The identification of location attributes is being performed by looking at specific prepositions and relations listed in Table 17 and Table 18, respectively.

Table 16. List of relations that indicate a temporal attribute.

Relations
[verb] + at + “time instance”
[verb] + at + [approximation] + “time instance”
[verb] + for + “time duration”
[verb] + for + [approximation] + “time duration”
[noun] + [verb] + “time duration”
[noun] + [verb] + [approximation] “time duration”

Table 17. List of keywords that indicate a location attribute.

Proximity	Located Above	Located Below
Across from	Above	Below
Adjacent	Anterior	Beneath
Alongside	Atop	Bottom
Approaching	Beyond	Deep
Beside	High	Down
Close	On top of	Down from
Close by	Over	Downward
Contiguous	Overhead	Low
Distant from	Upward	Posterior
In proximity		Under
Near		Underneath
Nearby		
Neighboring		
Next to		
Receding from		
Remote		
Retreating from		

3.3 NLP Analysis of Temporal Relation Between Events

Another class of textual data that can often be retrieved from NPPs includes IRs that report multiple events linked by temporal relations. As also indicated in Section 3.3.4, temporal relations can be both quantitative (e.g., an event has occurred two hours after another event) and qualitative (e.g., an event has occurred before another event).

Table 18. List of relations that indicate a location attribute.

Relations
[verb] + “location keyword” + noun
Subj + “location keyword” + obj

Note that a temporal relation does not necessarily imply a causal relation. In this respect, OPM models can reconstruct the causal relationship between events if additional ER data is available. For the scope of this report, we are following Moerchen’s process (2009), which lists the major temporal relations between events (see Figure 12):

- *Order*: sequential occurrence of events
- *Concurrency*: (almost) simultaneous occurrence of events from beginning to end
- *Coincidence*: temporal intersection of events

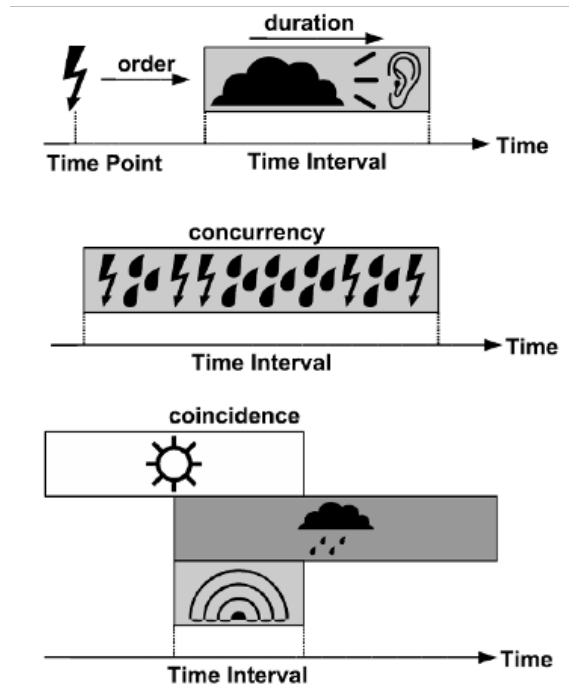


Figure 12. Graphical concepts of time-based relations: order, duration, concurrence, and coincidence of events (<https://archive.siam.org/meetings/sdm11/moerchen.pdf>).

Note that event duration (which is also indicated in Figure 12) does not provide information about temporal relations between events; instead, as indicated in Section 3.3.4, event duration is here considered a temporal attribute.

As described in Sections 3.2 and 3.3, the analysis of sentences containing temporal relations involves identifying specific keywords, relations, and grammatical structures in each sentence. In this respect, Table 19 and Table 20 provide the identified keywords (i.e., verbs, adjectives, and adverbs) and grammatical structures that indicate the order and coincidence of events.

Table 19. List of keywords and structures that indicate order of events.

Keywords			Structures
Verbs	Adjectives	Adverbs	
Antedate	After	Afterward	Soon after
Follow	Before	Consecutively	After that
Postdate	Consecutive	Consequently	After a while
Precede	Earlier	Directly	
Predate	Following	Hereafter	
Succeed	Former	Later	
	Later	Next	
	Next	Previously	
	Past	Subsequently	
	Precedent	Successively	
	Previous	Then	
	Prior	Thenceforth	
	Subsequent	Thereafter	
	Succeeding		
	Successive		

Table 20. List of keywords that indicate the concurrence and coincidence of events.

Keywords			Structures
Verbs	Adjectives	Adverbs	
Accompany	Accompanying	When	At that point
Conform	Attending	Thereupon	At that moment
Correspond	Coexistent	While	At that time
Harmonize	Concomitant	During	At that instant
Parallel	Concurrent		In the end
	Imminent		On that occasion
	Simultaneous		
	Synchronic		

Table 21. List of relations that indicate the order of events.

Relations
Event_1 + [order verb] + Event_2
Event_1 + [verb] + [adverb] + Event_2
Event_1 + [verb] + [adjective] + Event_2

Table 22. List of relations that indicate the concurrence and coincidence of events.

Relations
Event_1 + [verb] + [adverb] + Event_2
Event_1 + [verb] + [adjective] + Event_2

3.4 Examples of NLP Analysis

3.4.1 Rule-Based Health Information Extraction

We employ SSC OPM models (see Section 2.1) to generate a set of object-process language textual elements that lists not only all OPM elements but also their relationship. The textual OPM elements can be directly used by the Safety, Risk, Reliability Model Library (SR²ML) NLP module to identify these elements, that is the rule-based named entity recognition (NER), in the raw IR text. Before the rule-based NER process, the SR²ML NLP module performs several syntactic analyses on the raw text, as shown in Figure 13.

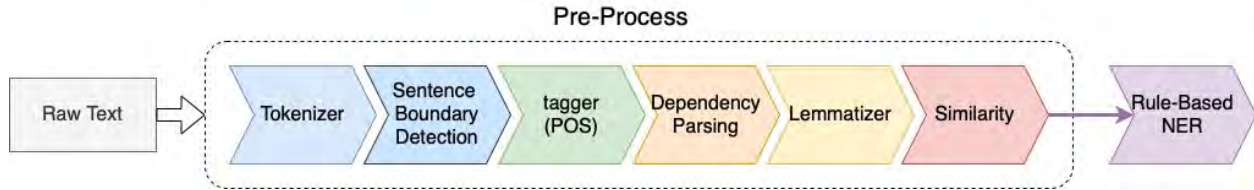


Figure 13. SR²ML NLP process.

In this research, we have collected a list of typical examples of IR descriptions (see Table 23) to test the effectiveness of the SR²ML NLP module. First, a list of SSCs tags is generated from the OPM model (see Table 24). These tags are used to formulate a list of patterns directly adopted by the SR²ML NLP module. The extracted entities and their health status are highlighted in blue and yellow, respectively, as presented in Table 25. In order to have a better illustration of the extracted data, we have presented the pair of SSC entities and health statuses in Table 26. As we observed, there are two misidentifications highlighted in green. They are easily resolved if we also include the health status keyword (highlighted in red) in the health status as illustrated in the following examples:

<i>Pump test failed due to power supply failure.</i>	<i>Pump, test → Pump, test failed</i>
<i>RCP pump 1A was cavitating and vibrating to some degree during test.</i>	<i>Pump, some degree → Pump, vibrating some degree</i>

Table 23. Examples for IR textual reports.

A leak was noticed from the RCP Pump 1A. RCP Pump 1A pressure gauge was found not operating. RCP Pump 1A pressure gauge was found inoperative. RCP Pump 1A had signs of past leakage. The pump is not experiencing enough flow during test. Slight vibrations noticed—likely from pump shaft deflection. Pump flow meter was not responding. Rupture of pump bearings caused pump shaft degradation. Rupture of pump bearings caused pump shaft degradation and consequent flow reduction. Power supply has been found burnout. Pump test failed due to power supply failure. Pump inspection revealed excessive impeller degradation. Pump inspection revealed excessive impeller degradation likely due to cavitation. Oil puddle was found in proximity of RCP Pump 1A. Anomalous vibrations were observed for RCP Pump 1A. Several cracks on the pump shaft were observed; they could have caused pump failure within few days. RCP Pump 1A was cavitating and vibrating to some degree during the test. This is most likely due to low flow conditions rather than mechanical issues. Cavitation was noticed but did not seem severe. The pump shaft vibration appears to be causing the motor to vibrate as well. The pump had noise of cavitation, which became faint after OPS bled off the air. Low flow conditions most likely causing cavitation. The pump shaft deflection is causing the safety cage to rattle. The pump is not experiencing enough flow for the pumps to keep the check valves open during the test. The pump shaft made noise. Vibration

seems like it is coming from the pump shaft. Visible pump shaft deflection. Pump bearings appear in acceptable condition. The pump made noises—not enough to affect performance. Pump shaft has a slight deflection.

Table 24. OPM elements identified in the IRs of Table 23 from the centrifugal pump OPM model shown in Figure 4.

External low-pressure water flow
Internal high-velocity water flow
External high-pressure water flow
Pump
Pump bearings
Pump shaft
Impeller
Diffuser
Housing
Casing
Centerline support
Seal
Motor
Pump coupling
Rotor
Stator
Power supply
Cooling system
Lubrication pump
Cooling fan

Table 25. Information extraction by SR²ML NLP module.

A leak was noticed from the RCP pump 1A. RCP pump 1A pressure gauge was found not operating. RCP pump 1A pressure gauge was found inoperative. RCP pump 1A had signs of past leakage. The Pump is not experiencing enough flow during test. Slight Vibrations is noticed - likely from pump shaft deflection. Pump flow meter was not responding. Rupture of pump bearings caused pump shaft degradation. Rupture of pump bearings caused pump shaft degradation and consequent flow reduction. Power supply has been found burnout. Pump test failed due to power supply failure. Pump inspection revealed excessive impeller degradation. Pump inspection revealed excessive impeller degradation likely due to cavitation. Oil puddle was found in proximity of RCP pump 1A. Anomalous vibrations were observed for RCP pump 1A. Several cracks on pump shaft were observed; they could have caused pump failure within few days. RCP pump 1A was cavitating and vibrating to some degree during test. This is most likely due to low flow conditions rather than mechanical issues. Cavitation was noticed but did not seem severe. The pump shaft vibration appears to be causing the motor to vibrate as well. Pump had noise of cavitation which became faint after OPS bled off the air. Low flow conditions most likely causing cavitation. The pump shaft deflection is causing the safety cage to rattle. The Pump is not experiencing enough flow for the pumps to keep the check valves open during test. Pump shaft made noise. Vibration seems like it is coming from the pump shaft. Visible pump shaft deflection. Pump bearings appear in acceptable condition. Pump made noises - not enough to affect performance. Pump shaft has a slight deflection.

Table 26. Extracted SSC entities and their health status.

SSC Entities	Health Status
Pump	A leak
Gauge	Not operating
Gauge	Inoperative
Pump	Signs of past leakage
Pump	Not enough flow
Pump shaft	Slight vibrations
Pump	Not responding
Pump bearings	Rupture
Pump shaft	Degradation
Pump bearings	Rupture
Pump shaft	Degradation
Power supply	Burnout
Pump	Test
Pump supply	Failure
Pump	Inspection
Impeller	Degradation
Pump	Inspection
Impeller	Degradation
Pump	Oil puddle
Pump	Anomalous vibrations
Pump shaft	Several cracks
Pump	Failure
Pump	Some degree
Pump shaft	Vibration
Motor	Vibrate
Pump	Noise of cavitation
Pump shaft	Deflection
Pump	Not enough flow
Pump shaft	Noise
Pump shaft	Vibration
Pump shaft	Deflection
Pump bearings	Acceptable condition
Pump	Noises
Pump shaft	A slight deflection

3.4.2 Rule-Based Causal Relation Identification

For the extraction of the causal relationship between SSCs in a sentence, we employ a set of rule templates based on specific trigger words and relations (see Section 3.1). Once the SSCs entities and their

health status has been identified, we can apply these rules to identify the causal relations. Using the same example presented in Table 23, we can identify the following causal relations presented in Table 27 after the rule-based NER process (the results are presented in Table 25). There is one causal relation that is not captured by SR²ML NLP module because “safety cage” is not listed in the OPM model. This can help us to enhance our OPM model.

Table 27. Causal relations identified by SR²ML NLP module.

Text After Rule-Based NER	Identified Causal Relations
<i>Rupture of pump bearings caused pump shaft degradation.</i>	(pump bearings: Rupture) “caused” (pump shaft: degradation)
<i>Rupture of pump bearings caused pump shaft degradation and consequent flow reduction.</i>	(pump bearings: Rupture) “caused” (pump shaft: degradation)
<i>Pump test failed due to power supply failure.</i>	(Pump: test failed) “due to” (power supply: failure)
<i>Pump inspection revealed excessive impeller degradation.</i>	(Pump: inspection) “revealed” (impeller: degradation)
<i>Pump inspection revealed excessive impeller degradation likely due to cavitation.</i>	(Pump: inspection) “revealed” (impeller: degradation)
<i>Several cracks on pump shaft were observed; they could have caused pump failure within few days.</i>	(pump shaft: Several cracks) “caused” (pump: failure)
<i>The pump shaft deflection is causing the safety cage to rattle.</i>	None

3.4.3 Coreference Handling

This process is tasked with finding the expressions that refer to the same entity in the text. This is particularly relevant where the text includes several sentences and a reference to an entity is not indicated with its proper name but with a pronoun. Through the SR²ML NLP module, we can correctly identify the coreferences in the text presented in Table 23 as shown in Table 28.

Table 28. Example of coreference identification.

Coreference Examples	Identified Coreference
<i>Several cracks on pump shaft were observed; they could have caused pump failure within few days.</i>	(Several cracks, they)
<i>Vibration seems like it is coming from the pump shaft.</i>	(Vibration, it)

3.5 NLP Methods Development

In this work, we have integrated SpaCy (<https://github.com/explosion/spaCy>), PySBD (<https://github.com/nipunsadvilkar/pySBD>), and Coreferee (<https://github.com/msg-systems/coreferee>) for text data analyses. SpaCy is an open-source Python library, released under the MIT license¹, for advanced NLP, including tagging, parsing, NER, text classification, and more. It features state-of-the-art speeds and provides a variety of linguistic annotations to give insights into a text’s grammatical structure. PySBD is a rule-based sentence boundary disambiguation Python package, released under the MIT

¹ <https://opensource.org/licenses/MIT>

license, to determine sentence boundaries. `Coreferee` is also an open-source Python library, released under the MIT license, to resolve coreferences. Table 29 presents a list of analysis steps we employed to process digital text data. In the following subsections, we present a more detailed description for each analysis step.

Table 29. NLP analysis pipelines.

ID	NLP Steps	NLP Pipeline	Note
1	Tokenization	tokenizer (SpaCy)	Segmenting text into words, punctuations marks, etc.
2	Sentence segmentation	pysbdSentenceBoundaries (PySBD)	Finding and segmenting individual sentences
3	Part of speech (POS)	tagger (SpaCy)	Assigning word types to tokens, like verb or noun
4	Dependency parsing	parser (SpaCy)	Assigning syntactic dependency labels and describing the relations between individual tokens, like subject or object
5	Lemmatization	lemmatizer (SpaCy)	Assigning the base forms of words, such as the lemma of “was” is “be” and the lemma of “pumps” is “pump”
6	Similarity	tok2vec (SpaCy)	Comparing words, text spans, and documents and how similar they are to each other
7	Rule-based entity recognition	PhraseMatcher, DependencyMatcher, and entity_ruler (SpaCy)	Finding sequences of tokens based on their texts and linguistic annotations and labeling named SSCs
8	Merge phrase or merge noun chunks	mergePhrase (customized)	Merging the noun with the words describing the noun as a single token
9	Coreference	initCoref (customized), Coreferee (Coreferee), anaphorCoref (customized)	Resolving coreference situations where two or more words within a text refer to the same entity

3.5.1 Tokenization

The first step in processing the text is to tokenize it using a SpaCy tokenizer (i.e., segment it into a list of words, punctuation, and so on) by applying rules specific to raw text, as illustrated by Figure 14. First, the raw text is split on whitespace characters. Then, the tokenizer processes the text from left to right. On each substring, it performs two checks:

1. Does the substring match a tokenizer exception rule? For example, “don’t” does not contain whitespace but should be split into two tokens, “do” and “n’t.”
2. Can a prefix, suffix, or infix be split off, such as punctuation like commas, periods, hyphens or quotes?

If there’s match, the rule is applied, and the tokenizer continues its loop starting with the newly split substrings. This way, tokenizer can split complex, nested tokens like combinations of abbreviations and multiple punctuation marks.

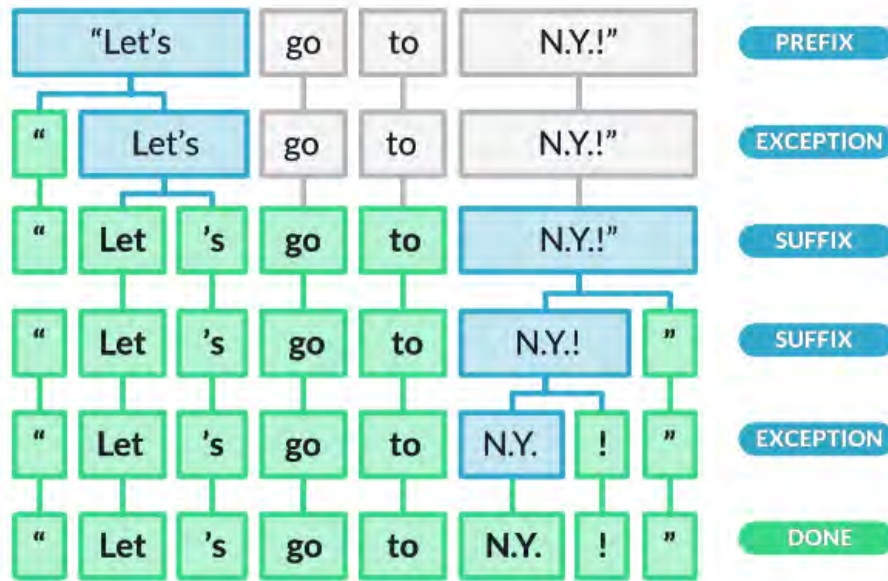


Figure 14. Tokenization process (source: <https://spacy.io/usage/spacy-101>).

3.5.2 Sentence Segmentation

The next important step is to determine the sentence boundaries, that is, segment the text into a list of sentences. It is a key underlying task for NLP process. In this work, we employ PySBD (<https://github.com/nipunsadvilkar/pySBD>), a rule-based sentence boundary disambiguation Python package, to detect the sentence boundaries. We have developed a custom pipeline using PySBD that is used together with SpaCy to split text into a list of sentences. In general, there are three different approaches to segment sentences: 1) rule based, requiring a list of hand-crafted rules, 2) supervised ML, requiring training datasets with labels and annotations, and 3) unsupervised ML, requiring distributional statistics derived from raw text. We choose the rule-based approach because the errors are interpretable and the rules can be adjusted incrementally. Moreover, the performance can be better than the ML models. For example, PySBD passes 97.93% of the Golden Rule Set exemplars (a language-specific set of sentence boundary exemplars) for English, with an improvement of 25% over the next best open-source Python tool (Sadvilkar and Neumann 2020).

3.5.3 Part of Speech

After the correct segmentation of sentences, we used SpaCy tagger to parse each sentence and tag each token in the sentence. Both "TAG" and "POS" attributes are generated for each token after the SpaCy tagger process. "POS" is the simple universal part-of-speech tag, does not include information for any morphological features, and only covers the word type (<https://universaldependencies.org/u/pos/>). The morphology is the process by which a root form of a word is modified by adding prefixes or suffixes that specify its grammatical function but do not change its POS. These morphological features are added to each token after the POS process and can be accessed through token's "morph" attribute. In addition, the "TAG" attribute expresses the POS and some amount of morphological information. For example, the POS "VERB" tag is expanded into six "TAG" tags, "VB" (verb, base form), "VBD" (verb, past tense), "VBG" (verb, gerund or present participle), "VBN" (verb, past participle), "VBP" (verb, non-3rd person singular present), and "VBP" (verb, 3rd person singular present). In this work, we employ these POS and TAG tags

to determine the description of the SSC health status (conjecture or qualitative observations). Table 30 presents the detailed English POS tags.

Table 30. Part-of-speech tags (source: <https://v2.spacy.io/api/annotation>).

TAG	POS	Morphology	Description
\$	SYM		symbol, currency
``	PUNCT	PunctType=quot PunctSide=ini	opening quotation mark
"	PUNCT	PunctType=quot PunctSide=fin	closing quotation mark
,	PUNCT	PunctType=comm	punctuation mark, comma
-LRB-	PUNCT	PunctType=brck PunctSide=ini	left round bracket
-RRB-	PUNCT	PunctType=brck PunctSide=fin	right round bracket
.	PUNCT	PunctType=peri	punctuation mark, sentence closer
:	PUNCT		punctuation mark, colon or ellipsis
ADD	X		email
AFX	ADJ	Hyph=yes	affix
CC	CCONJ	ConjType=comp	conjunction, coordinating
CD	NUM	NumType=card	cardinal number
DT	DET		determiner
EX	PRON	AdvType=ex	existential there
FW	X	Foreign=yes	foreign word
GW	X		additional word in multiword expression
HYPH	PUNCT	PunctType=dash	punctuation mark, hyphen
IN	ADP		conjunction, subordinating or preposition
JJ	ADJ	Degree=pos	adjective
JJR	ADJ	Degree=comp	adjective, comparative
JJS	ADJ	Degree=sup	adjective, superlative
LS	X	NumType=ord	list item marker
MD	VERB	VerbType=mod	verb, modal auxiliary
NFP	PUNCT		superfluous punctuation
NIL	X		missing tag
NN	NOUN	Number=sing	noun, singular or mass
NNP	PROPN	NounType=prop Number=sing	noun, proper singular
NNPS	PROPN	NounType=prop Number=plur	noun, proper plural
NNS	NOUN	Number=plur	noun, plural
PDT	DET		predeterminer
POS	PART	Poss=yes	possessive ending
PRP	PRON	PronType=prs	pronoun, personal
PRP\$	DET	PronType=prs Poss=yes	pronoun, possessive
RB	ADV	Degree=pos	adverb

RBR	ADV	Degree=comp	adverb, comparative
RBS	ADV	Degree=sup	adverb, superlative
RP	ADP		adverb, particle
SP	SPACE		space
SYM	SYM		symbol
TO	PART	PartType=inf VerbForm=inf	infinitival “to”
UH	INTJ		interjection
VB	VERB	VerbForm=inf	verb, base form
VBD	VERB	VerbForm=fin Tense=past	verb, past tense
VBG	VERB	VerbForm=part Tense=pres Aspect=prog	verb, gerund or present participle
VBN	VERB	VerbForm=part Tense=past Aspect=perf	verb, past participle
VBP	VERB	VerbForm=fin Tense=pres	verb, non-3rd person singular present
VBZ	VERB	VerbForm=fin Tense=pres Number=sing Person=three	verb, 3rd person singular present
WDT	DET		wh-determiner
WP	PRON		wh-pronoun, personal
WP\$	DET	Poss=yes	wh-pronoun, possessive
WRB	ADV		wh-adverb
XX	X		unknown
_SP	SPACE		

3.5.4 Dependency Parsing

POS provides information about word types and morphological features, but it does not provide the dependency information between words. We employ Spacy parser to label dependency parsing. Examples of dependencies include nominal subject (nsubj), direct object (dobj), indirect object (iobj), etc. The parser uses a variant of the non-monotonic arc-eager transition-system described by Honnibal and Johnson (2014). The parser uses the terms “head” and “child” to describe the words connected by a single arc in the dependency tree. The dependency labels, as listed in Table 31, are used for the arc label, which describes the type of syntactic relation that connects the child to head. Figure 15 shows an example with the graphic representation of the dependency tree using Spacy’s built-in displaCy visualizer, where the POS tag is placed below each word. In this work, we employ the dependency tree to develop rules for identifying SSCs’ health information and causal relationships between SSCs.

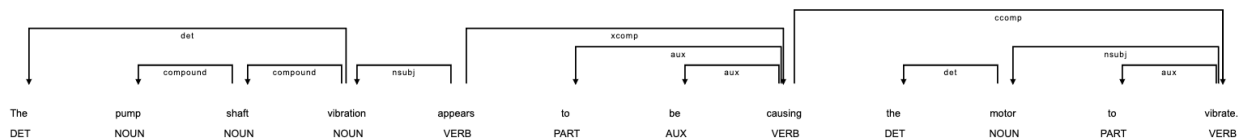


Figure 15. POS tagging and dependency parsing.

Table 31. Syntactic dependency parsing (source: <https://v2.spacy.io/api/annotation>).

Label	Description
acl	clausal modifier of noun (adjectival clause)
advcl	adverbial clause modifier
advmod	adverbial modifier
amod	adjectival modifier
appos	appositional modifier
aux	auxiliary
case	case marking
cc	coordinating conjunction
ccomp	clausal complement
clf	classifier
compound	compound
conj	conjunct
cop	copula
csubj	clausal subject
dep	unspecified dependency
det	determiner
discourse	discourse element
dislocated	dislocated elements
expl	expletive
fixed	fixed multiword expression
flat	flat multiword expression
goeswith	goes with
iobj	indirect object
list	list
mark	marker
nmod	nominal modifier
nsubj	nominal subject
nummod	numeric modifier
obj	object
obl	oblique nominal
orphan	orphan
parataxis	parataxis
punct	punctuation
reparandum	overridden disfluency
root	root
vocative	vocative
xcomp	open clausal complement

3.5.5 Lemmatization

A lemma is the base form of a token. The lemma of failing, fails, and failed is fail. Lemmatization is the process of reducing words to their base forms or lemmas. In this study, we employ the SpaCy lemmatizer to reduce inflectional forms or derivationally related forms of a word to a common base form. In this case, we only need to provide the base forms of keywords that leads to a significant reduction in the number of keywords.

3.5.6 Similarity

Another tool to reduce the number of keywords is to build a recommendation system utilizing word similarity. The similarity is determined by comparing word vectors or “word embeddings.” We employ SpaCy tok2vec pipeline and similarity attribute to compute similarity scores for making a prediction of how similar the keywords are.

3.5.7 Rule-Based Entity Recognition

In this work, we employ the OPM model to generate a set of SSCs and their relations. We utilize the set of SSCs to construct patterns that can be directly pass into the SpaCy “entity_ruler” to identify and label the SSCs as recognized entities. The “entity_ruler” is a SpaCy pipeline that lets us add named entities, such as OPM objects and processes, which makes it easy to combine rule-based and statistical NER for even more powerful pipelines. The “entity_ruler” will find matches in the text and label them using the specified pattern label. If any matches were to overlap, the pattern matching most tokens takes priority. If they also happen to be equally long, the match occurring first in the text is chosen. Entity patterns are dictionaries with two keys: “label,” specifying the label to assign to the entity if the pattern is matched, and “pattern,” the match pattern. The entity ruler accepts two types of patterns:

- Phrase patterns for exact matches (string)
{“label”: “SSC”, “pattern”: “pump”}
- Token patterns with one dictionary describing one token (list)
{“label”: “SSC”, “pattern”: [{“LOWER”: “pump”}, {“LOWER”: “shaft”}]}

The “entity_ruler” can also accept an “id” attribute for each pattern. Using the “id” attribute allows multiple patterns to be associated with the same entity.

3.5.8 Merge Phrase

In order to obtain more informative description about the SSCs’ health status, we have developed a custom SpaCy pipeline “mergePhrase.” The purpose of “mergePhrase” is to combine a noun plus the words describing the noun, such as “slight vibrations,” “not enough flow,” “past leakage,” or “oil puddle.”

3.5.9 Coreference

Coreference are situations where two or more words within a text refer to the same entity, such as “*Several cracks on pump shaft were observed; they could have caused pump failure within few days.*” In order to better understand the causal relationship among SSCs, resolving coreferences is an important task in this work. We employ Coreferee (<https://github.com/msg-systems/coreferee>), an open-source Python library, to resolve coreferences within English texts. It uses a mixture of neural network and programmed rules to identify potential coreference mentions. In this work, we have developed several

custom SpaCy pipelines to make Coreferee work seamlessly with SpaCy, which helps us identify causal relations among multiple sentences.

4. RELIABILITY MODELING

Two of the challenges of current plant reliability approaches are the ability to integrate plant health data and support decision-making. Condition-based data and diagnostic and prognostic information are in fact not considered in plant reliability models to inform system engineers on the most critical components. Currently, the propagation of quantitative health data from the component to the system level is a challenge given the diverse nature and structure of the data. On the other hand, plant reliability methods (which are typically based on fault trees or reliability block diagrams) can effectively propagate data from the component to the system level, but failure rate or probability values are an approximated integral representation of the past industrywide operational experience, which neglects the present component health status (e.g., diagnostic and condition-based data) and projection (when available from prognostic data).

Our first claim is that system reliability models should propagate health information from the component to the system and plant level in order to provide a quantitative snapshot of system and plant health and identify the most critical components. Our second claim is that component health should be informed solely by that specific component current and historical performance data and should not be an approximated integral representation of the past industrywide operational experience.

We are directly supporting these two claims by proposing a different approach for performing reliability modeling that relies on available component diagnostic, prognostic, and condition-based data to measure component health and propagates this information through fault-tree models. The propagation of health data from the component to the system level is performed not in terms of probability but in terms of margin, where margin is defined as the distance between the present actual status and an undesired event (e.g., failure or unacceptable performance). Through a cause-effect lens, while classical reliability models target the effect associated with a component performance, a margin-based approach focuses on the cause of an undesired component performance (i.e., component health). Hence, thinking of reliability in terms of margins implies decision-making based on causal reasoning. We show how fault-tree models can be solved using a margin language and how this process can effectively help system engineers identify the most critical components.

4.1 Summary of Margin-Based Reliability Modeling

Current reliability models are based on Boolean logic structures (Lee and McCormick 2011) (e.g., fault trees), which describe the deterministic functional relationship between SSCs and human interventions. Each basic event (BE) in a reliability model represents a specific elemental occurrence (failure of a component, failure to perform an action by the plant operators, recovery of a safety system, etc.), and a probability value is associated with each BE, which represents the probability that the BE can occur. However, maintenance and surveillance operations are typically not completely integrated into a probabilistic risk assessment structure. In addition, a probability value associated with an event is thus an integral representation of the past operational experience for such an event, and such value does not incorporate information on the present SSC health status (e.g., from diagnostic and condition-based data) and health projections (when available from prognostic data) on anticipated changes in SSC condition and performance in the near future.

A possible alternate path can start by redefining the word “reliability” to encompass a broader meaning that better reflects the needs of a system health and asset management decision-making process. Rather than focusing on how likely an event is to occur (in probabilistic terms), we think in terms of how far this

event is from occurring (Mandelli, Wang, and Hess 2021). This new interpretation of risk transforms the concept from one that focuses on the probability of occurrence to one that focuses on assessing how far away (or close) an SSC is to an unacceptable level of performance or failure. This transformation has the advantage that it provides a direct link between the SSC health evaluation process and standard plant processes used to manage plant performance (e.g., the plant maintenance and budgeting processes). The transformation also places the question into a form that is more familiar and readily understandable to plant system engineers and decision makers. When dealing with condition-based data (actual and archived data), margin \tilde{M} is defined here as the distance between observed past SSC conditions (e.g., oil temperature, vibration spectrum) that lead to failure.

Consider now two components, A and B . The margin \tilde{M} for both components can be visualized in a 2-dimensional space, as shown in Figure 16. Starting with brand-new components (i.e., $\tilde{M}_A, \tilde{M}_B = 1$), the aging degradation that affects both can be represented by the blue line in Figure 16, which parametrically represents the combination of the normalized margins ($\tilde{M}_A(t), \tilde{M}_B(t)$) as at a point in time t . Note that, if no maintenance (whether preventive or corrective) was ever performed on either component, this path would move from the coordinates (1,1) for Components A and B at the beginning of life to the coordinates (0,0) where both components had failed. We can identify these regions in Figure 16: the occurrence of both events where $\tilde{M}_A = 0$ and $\tilde{M}_B = 0$ and the occurrence of either event when $\tilde{M}_A = 0$ or $\tilde{M}_B = 0$. Now we can calculate \tilde{M} for the events listed above. This is accomplished by following the margin definition of the distance between the actual condition of Components A and B and \tilde{M} conditions identified by the event under consideration (e.g., the occurrence of both or either events):

$$\begin{aligned}\tilde{M}(A \text{ AND } B) &= \text{dist}[(\tilde{M}_A, \tilde{M}_B), (0,0)] \\ \tilde{M}(A \text{ OR } B) &= \min(\tilde{M}_A, \tilde{M}_B)\end{aligned}\tag{3}$$

The function $\text{dist}[X, Y]$ is designed to calculate the Euclidean distance between points X and Y .

Hence, exact solutions can be obtained extremely fast. More precisely, reliability calculations using \tilde{M} -based data can be performed by completing these four steps:

1. Construct the fault tree (FT); at this point, an FT contains only deterministic information about the architecture of the system under consideration (i.e., it simply models how the BEs are related to each other from a functional perspective).
2. Generate the minimal cut sets (MCSs) from the FT; as also indicated in Step 1, an MCS still represents the minimal combinations of BEs that lead to the top event.
3. Assign a margin value \tilde{M} to each BE.
4. Calculate the margin \tilde{M} of the union of the MCSs.

As part of system reliability modeling, it is always important to determine the importance of each BE. In a probabilistic risk assessment setting, this is performed by relying on risk-importance measures (RIM), such as Birnbaum or Fussell-Vesely. Given the different nature of \tilde{M} , it is possible to perform a risk-importance ranking by relying on a classical sensitivity measure (derivative based) for each BE defined as: $RIM_{BE} = \frac{\partial \tilde{M}(TE)}{\partial \tilde{M}(BE)}$. In other words, RIM_{BE} indicates how a small variation of $\tilde{M}(BE)$ directly affects the margin $\tilde{M}(TE)$ of the top event TE.

4.2 Notes on Distance Metrics

Note that, in a setting where margin values decrease as function of time, there is an infinite number of ways to move from the actual conditions of Components A and B , that is, the point $(\tilde{M}_A, \tilde{M}_B)$ to the point (0,0). However, the upper bound of such a distance is represented by the Manhattan metric,

$dist[\tilde{M}_A, \tilde{M}_B] = \tilde{M}_A + \tilde{M}_B$, where we note that there is no need to apply the absolute value convention as both \tilde{M}_A, \tilde{M}_B are by definition positive. On the other hand, the Euclidean metric represents the actual lower bound $dist[\tilde{M}_A, \tilde{M}_B] = \sqrt{\tilde{M}_A^2 + \tilde{M}_B^2}$. Thus, the metric distance should be selected based on the intended analysis, either conservative (Euclidean metric) or optimistic (Manhattan metric).

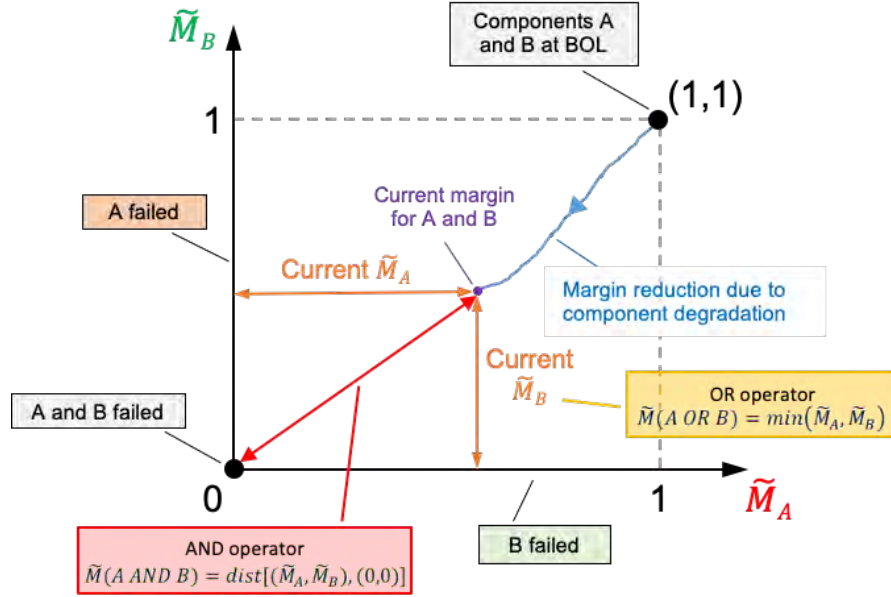


Figure 16. Graphical representation of event occurrences based on a margin framework.

Given these two bounds, which metric should be chosen? This question can be answered by referring to how condition-based data are collected. From a practical standpoint, condition-based data are collected from plant components on a regular basis (either continuously in time or at prescribed time intervals). Hence, at any specific time, we can quantify not only the margins for Components A and B (i.e., \tilde{M}_A, \tilde{M}_B) but also how these margin values change over time (i.e., $\frac{\partial \tilde{M}_A}{\partial t}, \frac{\partial \tilde{M}_B}{\partial t}$). The margin value for $\tilde{M}(A AND B)$ can now be better estimated by considering this new information.

An example of this process can be visualized with the graphics in Figure 17 where the margin value for both components uniformly decreases as a function of time (i.e., $\frac{\partial \tilde{M}_A}{\partial t}$ and $\frac{\partial \tilde{M}_B}{\partial t}$ are constant and do not change with time) but where the degradation of Component B occurs at a faster rate than Component A (i.e., $\frac{\partial \tilde{M}_A}{\partial t} < \frac{\partial \tilde{M}_B}{\partial t}$). In such conditions, the temporal evolution of \tilde{M}_A and \tilde{M}_B is represented by the continuous blue line of Figure 17. Starting from Point α (where components do not show any degradation, that is, brand new or recently refurbished to as good as new condition components), the blue line progresses up to actual measured conditions (Point β). Given the estimate of $\frac{\partial \tilde{M}_A}{\partial t}$ and $\frac{\partial \tilde{M}_B}{\partial t}$, it is now possible to estimate the progression of \tilde{M}_A and \tilde{M}_B in the future (i.e., the dashed blue line in Figure 17). This information can be used to estimate $\tilde{M}(A AND B)$ using base trigonometry rules as the length of the segment $\overline{\beta 0} = \overline{\beta \gamma} + \overline{\gamma 0}$. Note that this estimate of $\tilde{M}(A AND B)$ is still bounded by the Euclidean and Manhattan metrics but provides a more accurate estimate.

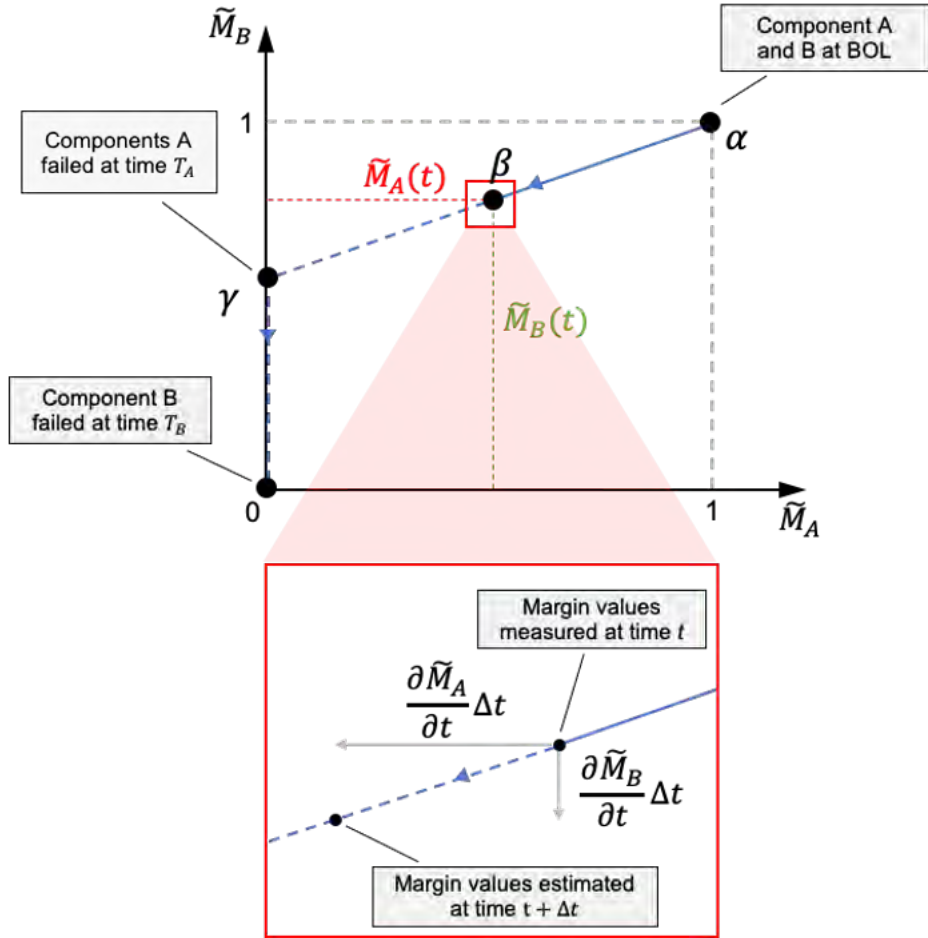


Figure 17. Graphical representation of the margin calculation for $\tilde{M}(A \text{ AND } B)$ when considering the temporal evolution of \tilde{M}_A and \tilde{M}_B .

Lastly, note that all these distance operations can be extended from a 2D to a generic n-dimensional space. In this analysis, once Point γ is reached (reflecting the condition where Component B has failed), estimating the time required to reach the point where both Component A and B failed now becomes a 1D problem, only dependent on the condition of Component A at the time of the Component B failure and the rate of degradation of Component A, which was linear in this simple example. It is noteworthy that the information used in the margin approach also provides the capability to estimate the time required to reach additional failure states that may be of interest for maintenance planning.

Note that historic information of $\frac{\partial \tilde{M}_A}{\partial t}$ and $\frac{\partial \tilde{M}_B}{\partial t}$ can be used to predict future \tilde{M}_A and \tilde{M}_B evolutions. As an example, Figure 18 shows the historic evolution of \tilde{M}_A and \tilde{M}_B and their predicted evolution based on the derivative information. The predicted evolution has been calculated by generating a random walk out of the distribution of $\frac{\partial \tilde{M}_A}{\partial t}$ and $\frac{\partial \tilde{M}_B}{\partial t}$. The same information can be plotted on the temporal scale, as indicated in Figure 19 where, given existing \tilde{M}_A and \tilde{M}_B data, the value of $\tilde{M}(A \text{ AND } B)$ using $\frac{\partial \tilde{M}_A}{\partial t}$ and $\frac{\partial \tilde{M}_B}{\partial t}$ is calculated (see the green line). The predicted evolution of $\tilde{M}(A \text{ AND } B)$ can be calculated given the predicted evolution of \tilde{M}_A and \tilde{M}_B (see the random walks plotted in red, blue, and purple for \tilde{M}_A , \tilde{M}_B and $\tilde{M}(A \text{ AND } B)$, respectively).

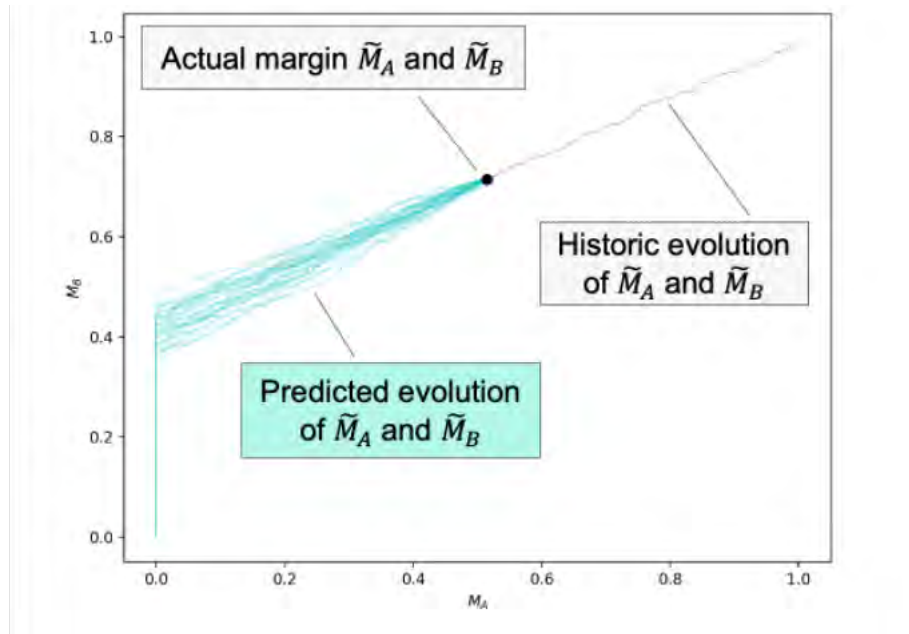


Figure 18. Plot of the historic evolution of \tilde{M}_A and \tilde{M}_B and their predicted evolution based on the derivative information.

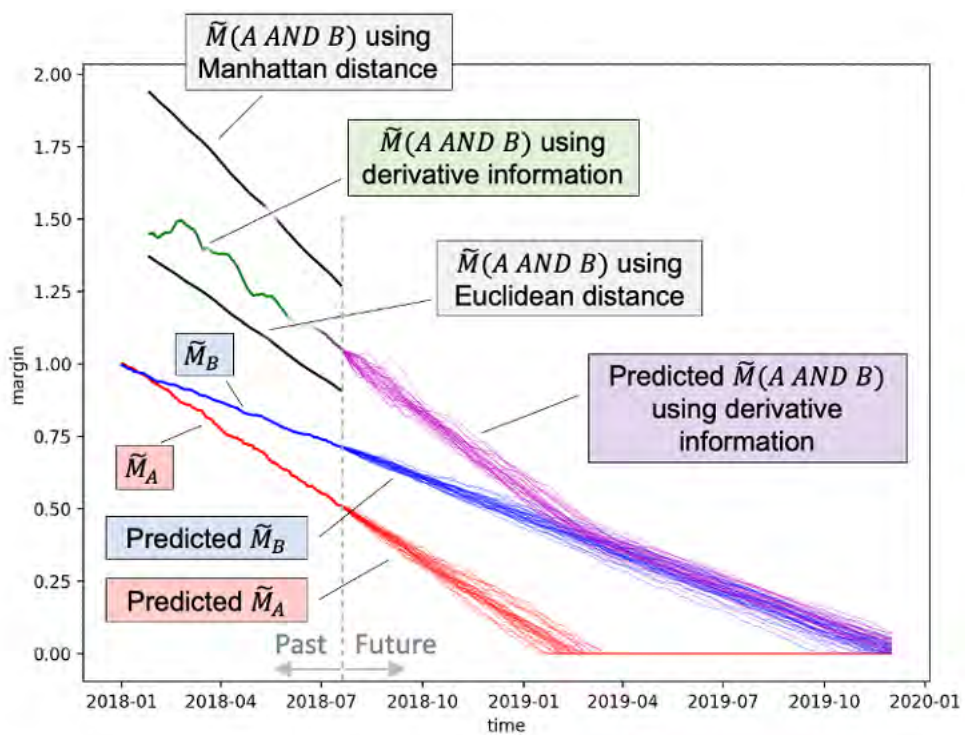


Figure 19. Plot of the historical and predicted evolution of \tilde{M}_A (red line), \tilde{M}_B (blue line), and $\tilde{M}(A \text{ AND } B)$ (green and purple line) based on the derivative information.

4.3 Margin Operations for AND, OR, KooN, and StandBy Operators

Section 4.1 has indicated how margin-based reliability approach can be propagated through AND and OR operators (see Equation 3). Note that those same expressions can be extended to multiple (i.e., more than two) BEs:

$$\begin{aligned}\tilde{M}(A \text{ AND } B \text{ AND } C) &= \text{dist}[(\tilde{M}_A, \tilde{M}_B, \tilde{M}_C), (0,0,0)] \\ \tilde{M}(A \text{ OR } B \text{ OR } C) &= \min(\tilde{M}_A, \tilde{M}_B, \tilde{M}_C)\end{aligned}\quad (4)$$

From a graphical point of view, the evolution of margin of N BEs can be plotted into an n -dimensional space where the same distance-based considerations shown in Section 4.1 still apply. The AND and OR operators are the basic elements that can be found in a system reliability model; however, two additional operators are typically found: the standby and the K out of N (KooN) operators.

The standby operator is designed to represent a configuration composed by two components that support the same function. At a given time, only one component supports such a function while the second one (on standby) is activated when the first fails or is taken out service. This operator is not too dissimilar from the AND operator; the main difference is that the margin of a component on standby does not change (i.e., there is no degradation) while the operating component is providing the desired function (see Figure 20). Note that this scenario is identical to the AND operator when the Manhattan distance is used:

$$\tilde{M}(A \text{ STAND-BY } B) = \tilde{M}_A + \tilde{M}_B \quad (5)$$

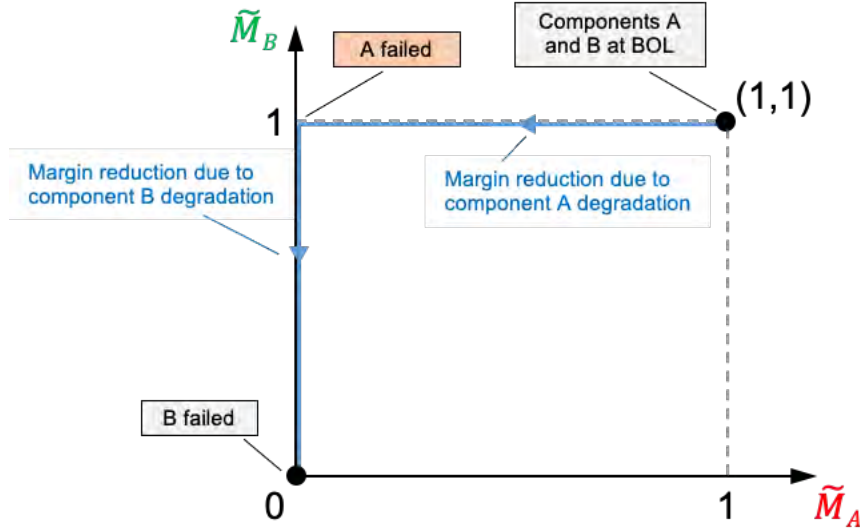


Figure 20. Margin evolution for two components, A and B, in a standby configuration.

The KooN configuration corresponds to a situation where out of N components that can support a specific function, only K (where $K < N$) of them are actually required. This situation is a convolution of both a parallel and series configuration. The margin calculation for this configuration can be deduced by following these considerations:

1. Let's consider all R combinations C_r ($r = 1, \dots, R$) of K components out of N
2. Each combination C_r is redundant in that all combinations are in a parallel configuration

3. The margin for each combination C_r is the minimum of the margin of its K components, that is, all K components in a configuration are in a series configuration
4. The minimum margin value of a combination C_r is the lowest margin of the K components
5. The maximum margin value of a combination C_r can be obtained by ranking in ascending order the margin of the K components and by considering the $N - K + 1$ highest

Hence, margin of a KooN configuration \tilde{M}_{KooN} can be obtained by (see Figure 21):

1. Ranking in ascending order the margin of the N components
2. Picking the first $N - K + 1$ components: $\tilde{M}_1, \dots, \tilde{M}_{N-K+1}$
3. Calculate \tilde{M}_{KooN} using the AND operator:

$$\tilde{M}_{KooN} = \text{dist}[(\tilde{M}_1, \dots, \tilde{M}_{N-K+1}), (0, \dots, 0)] \quad (6)$$

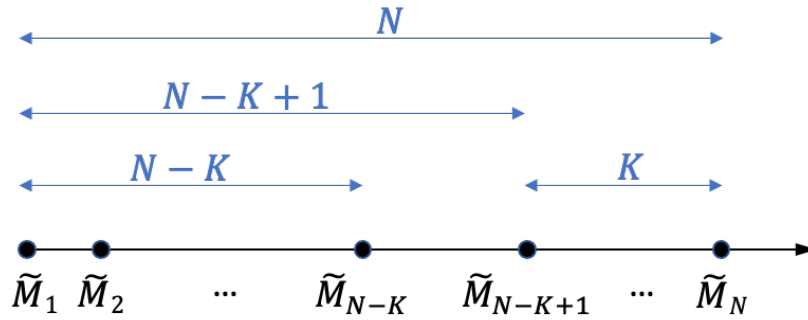


Figure 21. Margin calculation for KooN configuration.

Note that a series configuration corresponds to a KooN configuration where $K = N = 2$; Equation 6 would return the lowest margin, \tilde{M}_1 (see Section 4.1). A parallel configuration of Components A and B corresponds to a KooN configuration where $K = 1$ and $N = 2$. In this scenario, Equation 6 would return $\tilde{M}(A \text{ AND } B) = \text{dist}[(\tilde{M}_A, \tilde{M}_B), (0, 0)]$ (see Section 4.1).

4.4 Integration of ER Data

This section provides practical examples on how margin values can be estimated from available ER data. Given the variety of ER data, we have partitioned these examples into three categories: anomaly detection data (see Section 4.4.1), condition-based data (see Section 4.4.2), and prognostic data (see Section 4.4.3).

4.4.1 Anomaly Detection Data

Anomaly detection methods (Nassif 2021) are designed to identify unexpected behaviors, that is, outside the normal operation boundaries. Hence, they provide binary information about the status and health of a particular component (it either works normally or abnormally). In a margin context, an anomaly identifies an unexpected behavior (or an unknown failure model) that requires immediate attention. In this scenario, the quantification of a margin value from an anomaly detection method can be as follows:

$$\tilde{M} = \begin{cases} 1 & \text{Component working normally} \\ 0 & \text{Component under anomalous behavior} \end{cases} \quad (7)$$

Note that such an anomaly can be triggered by an internal (e.g., degradation and rupture of an internal part) or external (e.g., failure of another component that support a function the monitored component) event.

The binary nature of the margin definition provided above can be fairly limiting in practical situations. Depending on the employed anomaly detection method, it is possible to overcome this limitation by adapting the definition of margin on the detection computational engine. Within the RIAM project, we employ auto-associative kernel regression (AAKR) (Baraldi et al. 2015) for testing purposes due to its robustness and explainability advantages. The AAKR method employs historic measured data under normal conditions and validates this data set with currently measured data through a kernel regression (Mohri and Rostamizadeh 2012). Based on the observed data (Ξ^{obs-nc}) and currently measured data (Ξ^{obs}), the AAKR algorithm reconstructs the expected data values (Ξ^{rec}) through a regression:

$$\Xi^{rec} = \frac{\sum_{n=1}^N w(n) \Xi^{obs-nc}(n)}{\sum_{n=1}^N w(n)} \quad (8)$$

where:

$$w(n) = \frac{1}{\sqrt{2\pi h^2}} e^{-\frac{(\Xi^{obs} - \Xi^{obs-nc}(n))^2}{2h^2}} \quad (9)$$

Under normal conditions, Ξ^{rec} should be very similar to Ξ^{obs} (i.e., $\Xi^{obs} \cong \Xi^{rec}$). The condition $\Xi^{obs} \neq \Xi^{rec}$, indicates anomalous behavior. Under the AAKR context, the same regression method can be employed to determine the component margin as follows:

$$\tilde{M} = 1 - \frac{\|\Xi^{obs} - \Xi^{rec}\|}{h} \quad (10)$$

By using Figure 22 as reference case, note that:

- If the component is under normal conditions (i.e., kernel regression is located within the component normal condition data Ξ^{obs-nc} , see the green points in Figure 22), $\Xi^{obs} \cong \Xi^{rec}$; hence $\tilde{M} = 1$
- If the component is under abnormal conditions, the norm of difference between Ξ^{obs} and Ξ^{rec} can be at most h .

4.4.2 Condition-Based Data

As indicated in Section 4.1, the margin can be calculated as the distance between actual and limiting conditions. In practical settings, limiting conditions can be represented by technical specifications specific to the component.

As an example, for induction motors, oil viscosity must be below a specified limiting condition to ensure proper motor function. Oil viscosity can significantly change as a function of motor rotation speed. Hence, the margin can be calculated as the difference between the specified limiting condition and the currently measured oil viscosity. In general terms, given an upper limiting condition x_{LC} for a monitored variable x_{obs} , the margin \tilde{M} can be defined as:

$$\tilde{M} = \frac{x_{LC} - x_{obs}}{x_{LC} - \min(x_{obs})} \quad (11)$$

where $\min(x_{obs})$ indicates the minimum allowable value for x_{obs} .

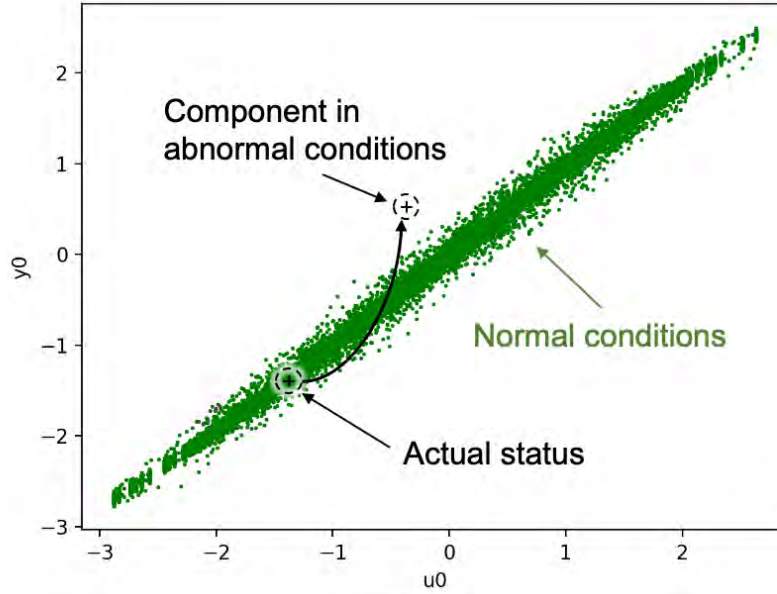


Figure 22. Integration of anomaly detection data into margin calculation (AAKR test case).

In a more practical setting, cage winding issues for three-phase induction motors may emerge within few years due to premature aging (Stone et al. 2014) and is usually caused by the degradation of the electrical insulation in the rotor (if present) and stator windings. In this context, we used current signature analysis testing to detect cage winding issues. This is performed by identifying two sideband currents centered around frequency f_2 :

$$f_2 = s f_1 \quad (12)$$

where f_1 indicates the supply frequency (i.e., 60 Hz) and s indicates slip factor. The two sideband currents f_{sb} are located at (see Figure 23):

$$f_{sb} = f_1 \pm 2s f_1 \quad (13)$$

If significant sideband currents $I_{f_{sb}}$ are present (dB difference between current I_{f_1} at f_1 and average sideband f_{sb} height ≤ 45 dB), cage winding breaks are likely to occur. Given this information, the margin can be defined with (see Figure 24):

$$\tilde{M} = \frac{(I_{f_1} - I_{f_{sb}}) - 45}{I_{f_1} - 45} \quad (14)$$

For rotating components (e.g., centrifugal pumps), a typical degradation process affects pump mechanical seals. The pump vibration signal is constantly monitored using standard accelerometers and data for normal and failed conditions (for example see Figure 25) are often available from manufacturers. Statistical indicators, such as the root mean square (RMS), of the signal can be measured. Examples of RMS analyses observed when seals are degraded beyond their limit for different pump rotation speeds are shown by Luo et al. (2021) and in Figure 26. In this context, margin \tilde{M} can be defined as:

$$\tilde{M} = \frac{T_{damaged} - T_{obs}}{T_{damaged} - T_{normal}} \quad (15)$$

where T represents the RMS value measured under difference conditions: normal, damaged, and observed.

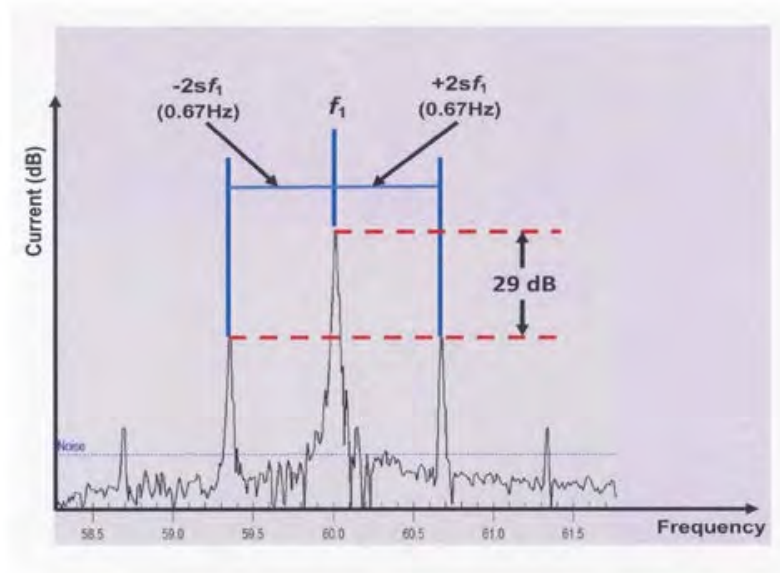


Figure 23. Typical current signature analysis where sideband currents $I_{f_{sb}}$ are centered around the frequency of the supply current. Source: <https://irispower.com/learning-centre/relative-merits-off-line-line-testing-rotating-machine-stator-rotor-windings/>.

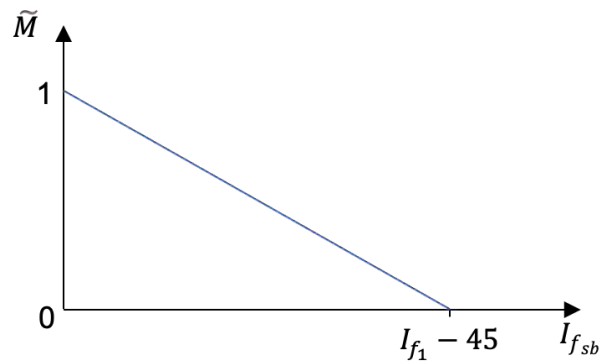


Figure 24. Graphical representation of margin provided measured sideband currents $I_{f_{sb}}$.

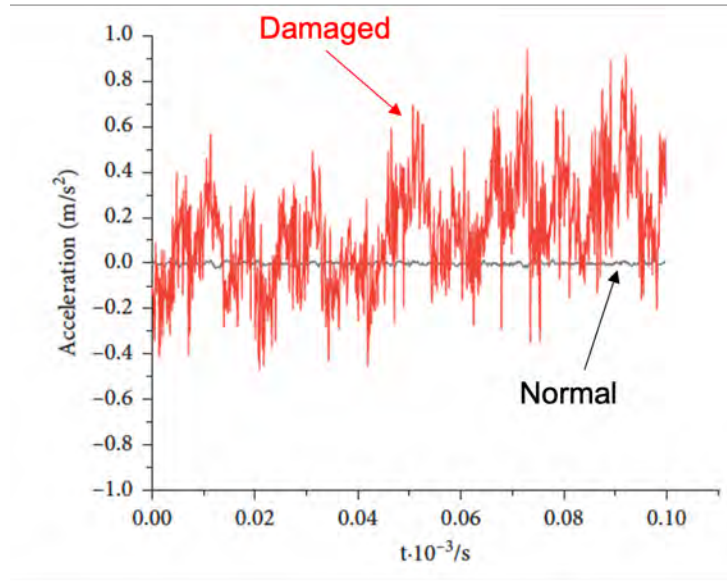


Figure 25. Time domain analysis of vibration signal. Source: Luo et al. (2021).

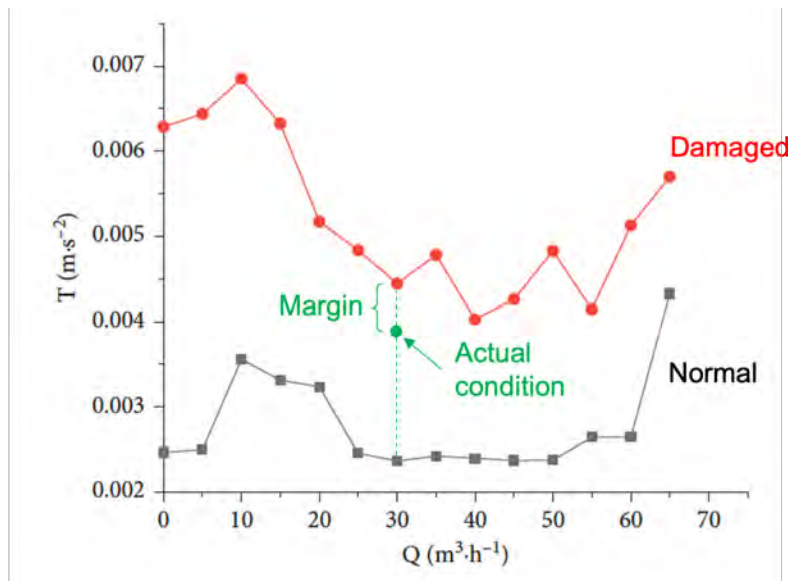


Figure 26. Vibration data for different mass flow rates and margin representation provided actual pump conditions. Adapted from Luo et al. (2021).

A more complex scenario can occur when normal and failed data is distributed over multiple variables. As an example, Figure 27 shows data for both normal and failed conditions distributed over two monitored variables (i.e., u_0 and y_0). This scenario extends the analysis presented in Figure 22 where both normal (indicated as Ξ_{NC}) and failed (indicated as Ξ_{fail}) data is available. In order to determine the margin, we rely on the ML concept of a classifier.

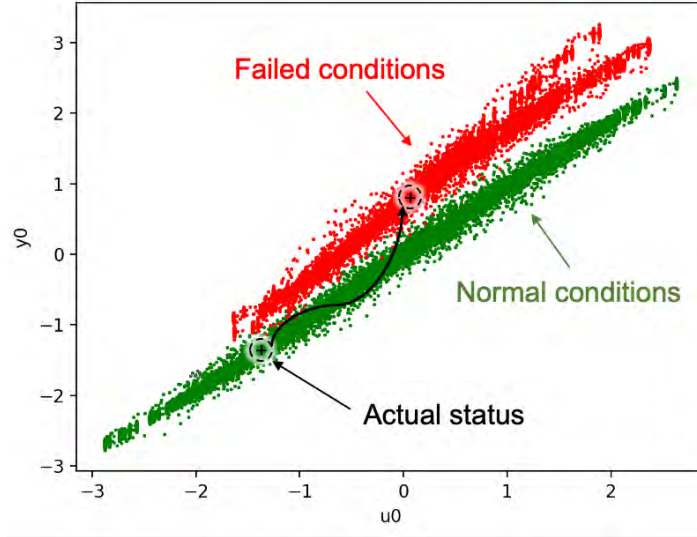


Figure 27. Example of scenario where normal and failed conditions (defined over a 2D space) are available to determine component margin.

As an example, if the K nearest neighbor classifier is employed, we can determine the margin \tilde{M} associated with actual measured data point P_{obs} by selecting the K closest data points P_k in the $\Xi_{NC} \cup \Xi_{fail}$ dataset:

$$\tilde{M} = \frac{\sum_{k=1}^K w(k) \cdot \delta_k}{\sum_{k=1}^K w(k)} \quad (16)$$

where

$$\delta_k = \begin{cases} 1 & \text{if } P_k \text{ is a normal condition data point (i.e., } P_k \in \Xi_{NC}) \\ 0 & \text{if } P_k \text{ is a failed condition data point (i.e., } P_k \in \Xi_{fail}) \end{cases} \quad (17)$$

and

$$w(k) = \frac{1}{dist^2(P_{obs}, P_k)} \quad (18)$$

As a practical example, a common failure mode associated with an induction motor is the failure of ball bearings that support shaft rotation. A database for both normal and faulty bearing conditions can be found at the Bearing Data Center of Case Western Reserve University². Researchers there conducted a set of experiments inserting bearing faults and collecting vibration data using accelerometers at both the drive end (DE) and fan end (FE) of the motor housing. Figure 28 plots the vibration data recorded at the DE and FE under normal conditions and when two faults have been inserted (with fault diameters, FD, set to 0.007 in. and 0.040 in.).

Note how this data set is structurally different from the one plotted in Figure 27 since the normal condition data is actually fully contained within failed data. The approach presented above and described by Equations 16–18 can still be applied by preprocessing the data and setting the weight $\delta_k = 1$ for failed data points within the envelop of the baseline (i.e., normal conditions data).

² Official website: <https://engineering.case.edu/bearingdatacenter>.

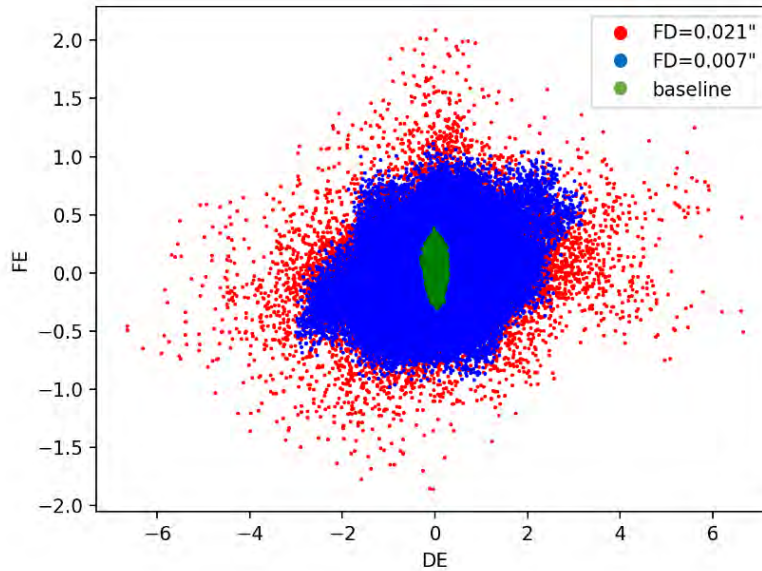


Figure 28. Vibration data recorded at the DE and FE under normal conditions and when two faults have been inserted (FD set to 0.007 in. and 0.040 in.). Source: <https://engineering.case.edu/bearingdatacenter>.

4.4.3 Prognostic Data

For components where a prognostic analysis is available, it is possible to estimate the component's remaining useful life (RUL) when component degradation starts to emerge. Typically, RUL is quantified in probabilistic terms where a probabilistic distribution function defined over the time axis t is generated, $RUL \sim PDF_{RUL}(t)$. In such a case, the margin can be estimated using two approaches. The first defines the margin as $M = 1 - CDF_{RUL}(t)$ where CDF_{RUL} indicates the cumulative distribution function corresponding to PDF_{RUL} . The second approach estimates the margin as the distance between the actual component life and a point estimate of the RUL distribution (e.g., the 5th percentile).

A graphical representation of the margins for both approaches is shown in Figure 29 for an estimated RUL normally distributed as shown in red. Note that the proposed approach updates the margin value when component health is measured and when a better RUL estimation (i.e., less uncertainty associated with RUL) is available from the corresponding prognostic model.

4.5 Notes on Common Cause Failures

In classical reliability modeling, an important element to consider is the notion of component failure induced by a common cause, that is, common-cause failures (CCFs). The probabilistic treatment of CCFs is a challenge, and several methods have been developed as described by Lee and McCormick (2011). The reason behind it is that if a CCF has been identified (e.g., from previous operational experience), the assumption of independence between events is no longer valid; this has strong implications when probability calculations are performed. As an example, let's consider two events, A and B, the goal is to determine then $P(A \text{ OR } B) = P(A) + P(B) + P(A \text{ AND } B)$ where $P(A \text{ AND } B) = P(A|B)P(B)$. If these events are independent, $P(A \text{ AND } B) = P(A)P(B)$. However, if common causes have been identified, the formulation of $P(A \text{ AND } B)$ cannot be simplified as $P(A)P(B)$ but needs to model the effect of CCFs through the term $P(A|B)$. This implies that $P(A \text{ OR } B)$ is calculated as $P(A) + P(B) + P(A)P(B) +$

$P(CCF)$ where $P(A)$ and $P(B)$ are quantified by neglecting the contribution due to CCFs and the effect of CCFs is isolated in the term $P(CCF)$. The challenge is the estimation of $P(A)$ and $P(B)$ (where CCF effects are taken out) along with $P(CCF)$ from available data such that the effects of common causes are not counted twice (in both $P(A)$ and $P(B)$).

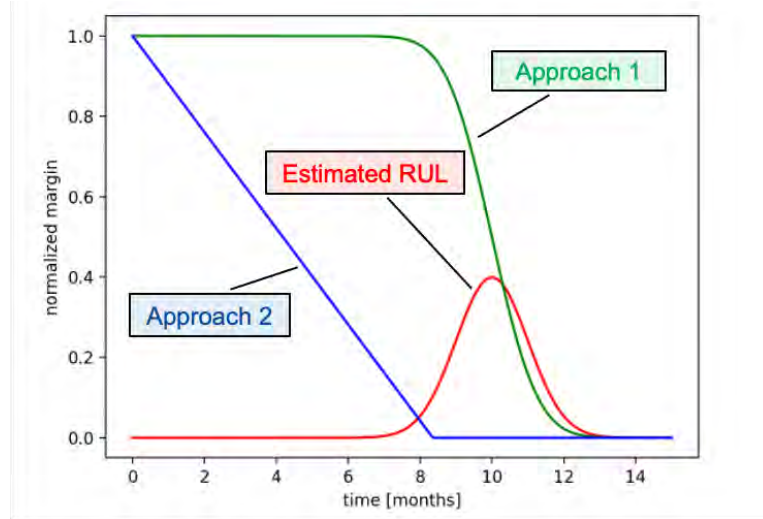


Figure 29. Margin values obtained from the two proposed approaches (green and blue lines) given an estimate of component's RUL (red line).

In a margin-based reliability setting, common causes still play a relevant role since external conditions might change the margins of multiple components or a causal relation between component margins might be present. However, we will show how the effect of common causes does not pose a challenge in a margin-based calculation and there is no need to isolate the effects induced by common causes to avoid double counting by considering the cause-effect diagram presented in Figure 30. This situation considers two components, A and B, with their health affected by two causes (Cause 1 for Component A and Cause 2 for Component B). A third cause, the common cause, affect both components. The health of Component A (and, hence, also its margin) includes the contribution of Cause 1 and the common cause; similarly, the health of Component B includes the contribution of Cause 2 and, again, the effect of the common cause. Recall that, when considering the margin for failure of either or both components (by solving $M(A \text{ OR } B)$ and $M(A \text{ AND } B)$, respectively, see Section 4.3), the concept of independence has never been mentioned. Whether common causes are present or not, the distance operations indicated in Section 4.3 require a full estimate of the margins $M(A)$ and $M(B)$, which includes the contribution of Cause 1, Cause 2, and the common cause. Hence, the effect of common causes must be accounted twice for both $M(A)$ and $M(B)$.

4.6 Examples of Margin-Based Reliability Calculations

A margin-based reliability calculation example is indicated here for the system shown in Figure 31 (Youngblood 2001) composed of seven components, A–G. For each component, an estimation of its RUL is provided (see the top plot of Figure 32). The RUL estimation is represented here probabilistically, that is, the RUL is represented by a probabilistic distribution function designed to represent the uncertainty associated with a RUL estimate (in terms of the RUL mean and variance). In this scenario, we represent system reliability in terms of minimal path sets (MPSs) rather than MCSs. A detailed reliability modeling

of the system shown in Figure 31 is provided in Appendix A. The system margin was calculated by considering the MPSs of the system of Figure 31 and applying the margin rules indicated in Section 4.3 using the following settings:

- Component RUL margin model: Approach 1 of Figure 29
- Distance metrics for $M(A \text{ AND } B)$: Euclidean

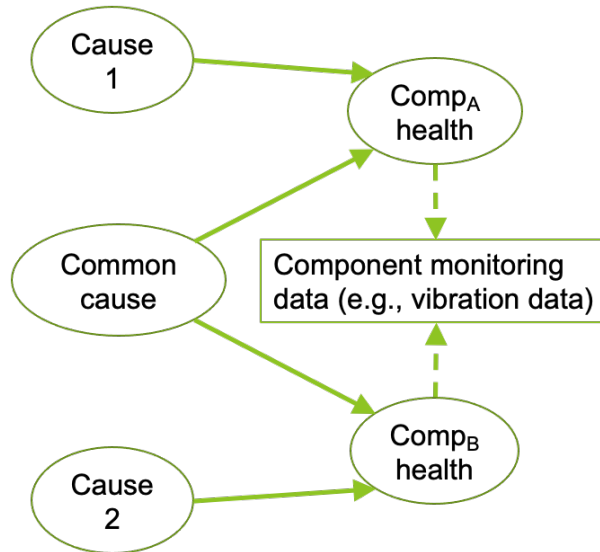


Figure 30. Graphical representation of common causes in a margin context.

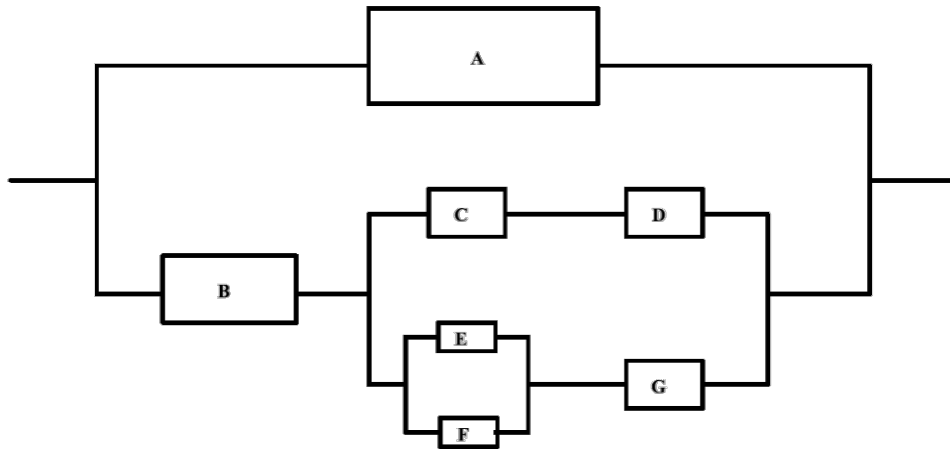


Figure 31. Example of system architecture represented in terms of block diagrams (Youngblood 2001).

The obtained temporal profile of system margin is shown in the bottom plot of Figure 32. From this plot, note the following:

1. Even though component margin is defined in the $[0,1]$ interval, the system margin can be higher than one (but still cannot be negative). A system margin greater than one implies that there are redundancies that can compensate for component failures. In other terms, when there is more than

one single MPS, the system margin is greater than one. At time $t = 0$, there are four MPSs, and the margin for each component is set to one, hence the system margin can be calculated as $\sqrt{1 + 1 + 1 + 1} = 2.0$.

2. When components are approaching their own RUL, their margin decreases to zero until they are considered failed. Hence the number of available MPSs decreases and system margins decreases as well to a value equal to the square root of the number of available MPSs.
3. At time $t = 8 \text{ months}$, Component E fails, and even though Components B and G are working properly, there are no available MPSs and, consequently, the system margin value drops to zero.

The next step is the quantification of the reliability measures (RIM) indicated in Section 4.1. Given the component margin values at each time instant and the obtained system margin (see Figure 32), these measures were calculated and plotted in Figure 33 for all seven components.

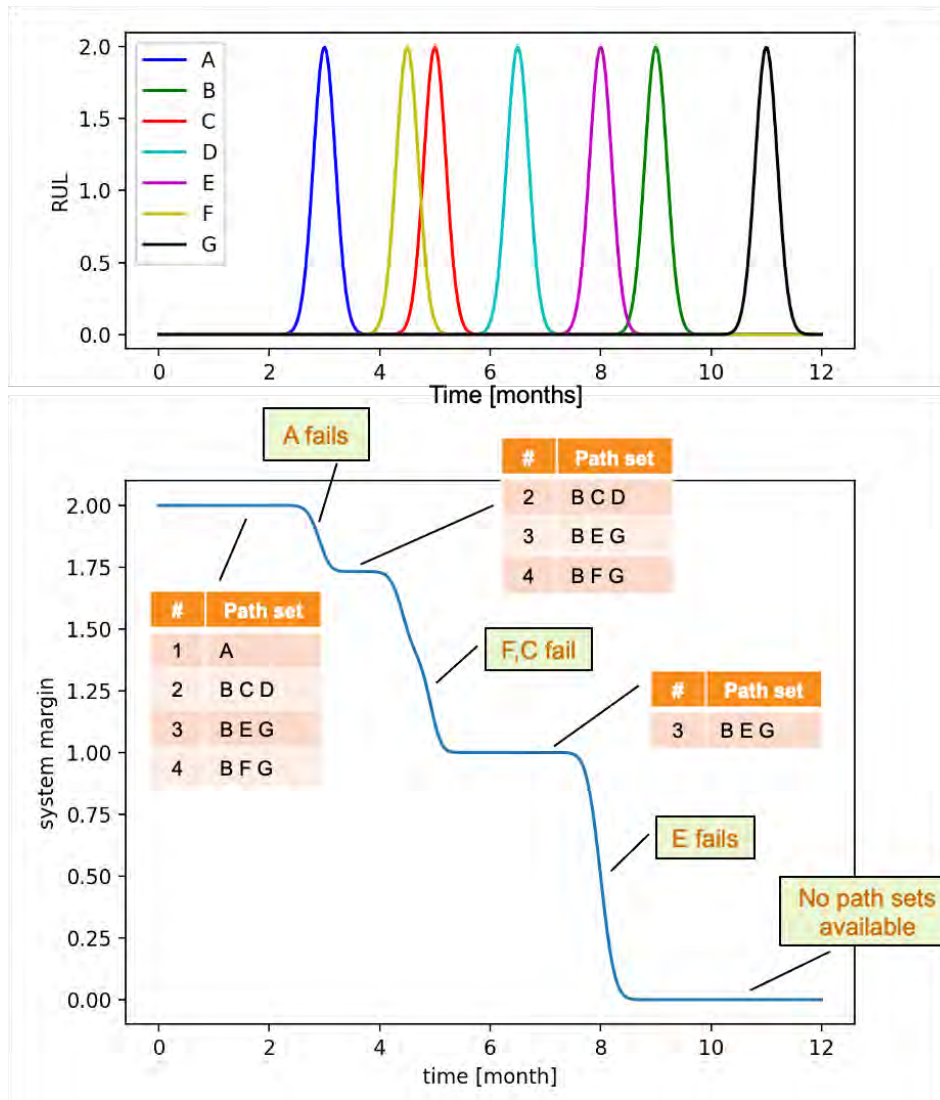


Figure 32. Example of margin-based calculations using prognostic data for the system indicated in Figure 31. For each of the seven components, A–G, a RUL is provided in the top plot and the corresponding system margin quantification is indicated in the bottom plot.

In order to show the type of information generated by these plots, let's consider the two regions of these plots highlighted in red:

1. The first region is located at the beginning of the plot where all components have a margin value of one (i.e., all components are healthy). However, note that the RIM values are not equal among the seven components due to the fact that each component directly supports a different number of MPSs. As an example, Component B supports three MPSs (BCD, BEG, BFG) while Component A supports one single MPS (A). Hence, improving the margin of Component B is more beneficial at the system margin.
2. The second region is located toward the end of the time axis where only one MPS is available (BEG). In this case, Components B and G are still healthy (i.e., their margin is still one) while Component E is approaching its RUL. Thus, the importance of Component E is greater than the importance of Components B and G.

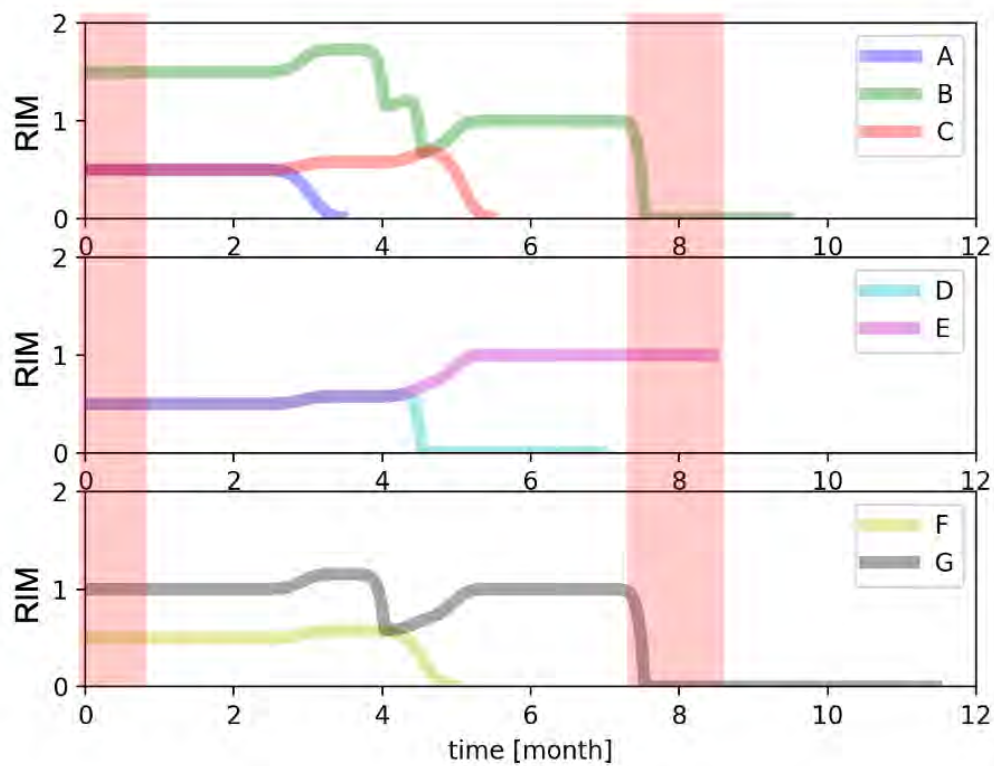


Figure 33. Plot of RIM measures for the seven components of the system of Figure 31 given the provided prognostic data (in terms of RUL) indicated in the top plot of Figure 32.

4.7 Links Between Margin and Classical Reliability Approaches

At this point, it is relevant to present the structural differences between classical reliability models (i.e., based on probability of failure) and a margin-based approach. These differences can be described through a cause-effect lens, as shown in Figure 34. Classical reliability models focus on the effect node (i.e., to model component failure), whereas component reliability data are used to assess the system failure probability. Such models are used to monitor plant risk (as currently done by plant risk monitors) and to set “offline” decisions, such as setting periodic surveillance and maintenance activities or to set the duration of planned

system maintenance outages (either as part of a plant configuration risk management program or a plant risk-managed technical specification program).

Over the past several decades, plants have been moving from a reliance on periodic to more comprehensive predictive strategies where the goal is to only perform intrusive maintenance operations when needed. Advanced monitoring and data analysis technologies are essential to support predictive strategies. This is where margin-based reliability approaches can be applied to support this type of “online” decision-making where, based on current condition-based data, component health data are employed to assess component and system health (i.e., the focus is now shifted to the cause node of Figure 34).

As a final remark, note the following:

- As indicated in Section 4.4, note that the margin value of a component reflects the status of a component provided actual monitored data
- The temporal evolution of a margin value implicitly includes all non-linear behaviors behind component degradation
- Sudden component performance degradation is mirrored by an equivalent step decrease of the corresponding margin value

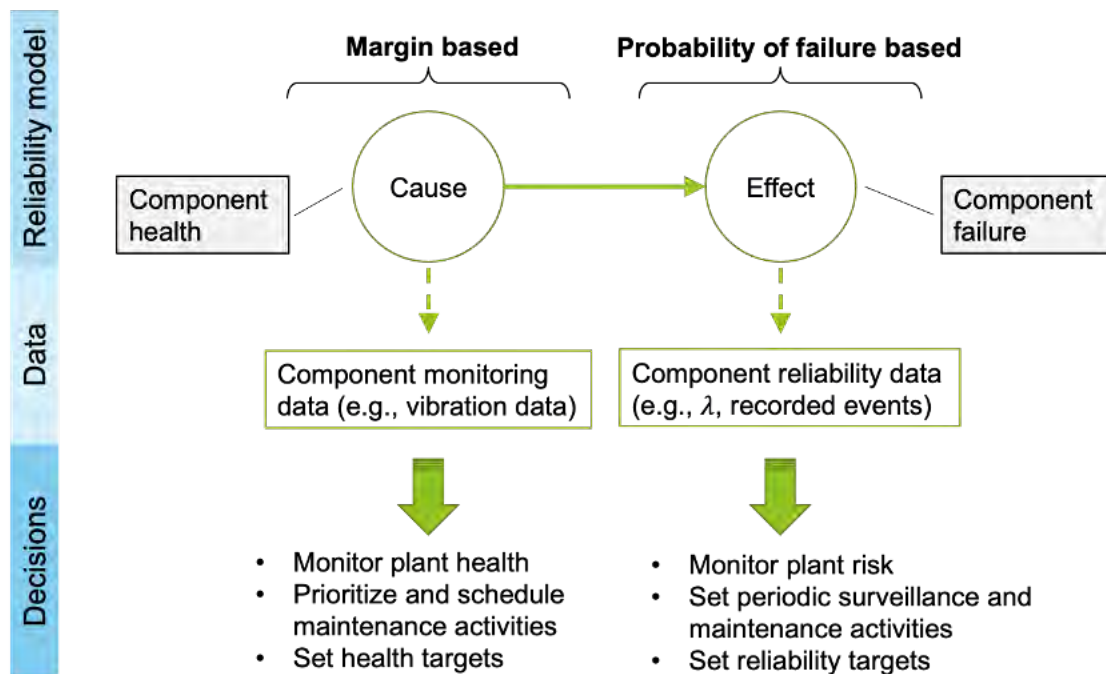


Figure 34. Comparison between margin-based and probability of failure-based reliability modeling approaches.

4.8 Numerical Comparison Between Margin and Classical Reliability Approaches

This section provides a more concrete bridge between the margin and classical reliability approaches in support of the statements provided in Section 4.7. A common ground for these two kinds of reliability

approaches can be established if ER data is composed solely of component failure time values. In this scenario, we are considering the failure time value for each component to determine the system failure time for the most common system configurations (see also Appendix A): series, parallel, standby, and KooN.

Given the failure rates for the considered components, the corresponding mean time to failure (MTTF) values are calculated. Using the set of equations provided in Appendix A, we can determine the system failure time of the considered system configurations using a classical reliability theory basis (see the central column of Table 32).

Similarly, component MTTF values can be translated in terms of margins to determine the system margin for the considered system configurations using the equations provided in Section 4.3. In this situation, a margin is basically the distance between the actual component life and its predicted life (i.e., the component's MTTF). The system margin M_{sys} for considered system configurations are presented in the right column of Table 32.

The goal now is to translate M_{sys} values back into time values and compare them to T_{sys} . We expect that this comparison process will provide identical outcomes from classical and margin reliability methods. At a first look, the mathematical expressions for T_{sys} and M_{sys} for the series, standby, and KooN configurations are very similar. Recall that components margin values (e.g., M_1, M_2 in Table 32) are defined over the time axis and quantified based on the component MTTF value; hence, the translation from M_{sys} to system failure time in a margin-based reliability context gives these outcomes:

- *Series*: given that M_{sys} is defined over a minimum of two component margins, the system failure time derived from the margin calculation corresponds to $\min(T_1, T_2)$ (see Figure 35).
- *Parallel*: in this configuration M_{sys} follows the path shown in Figure 36 where components fail at two different time instances while the system fails when the latter failure occurs (i.e., $\max(T_1, T_2)$).
- *Standby*: in this configuration, the sum of two margin values (i.e., $M_1 + M_2$) is translated into the sum of the failure time of both components (i.e., $T_1 + T_2$). Note that, in this configuration, $M_{sys} = M(A \text{ AND } B)$ where the L1 norm is used $M_{sys} = M_1 + M_2$ (see Figure 37).
- *KooN*: similar to the series configurations, only K components with the highest margin values are considered (and consequently the highest failure times). The minimum out of this K value gives the same outcome as $\min(\text{first } K \text{ highest } T_i)$.

Table 32. Comparison between system failure time T_{sys} and system margin M_{sys} for four system configurations.

Configuration	System Failure Time (classical reliability—see Appendix A)	System Margin (see Section 4.3)
Series of two components	$T_{sys} = \min(T_1, T_2)$	$M_{sys} = \min(M_1, M_2)$
Parallel of two components	$T_{sys} = \max(T_1, T_2)$	$M_{sys} = \text{dist}[(0,0), (M_1, M_2)]$
Standby	$T_{sys} = T_1 + T_2$	$M_{sys} = M_1 + M_2$
KooN	$T_{sys} = \min(\text{first } K \text{ highest } T_i)$	$M_{sys} = \min(\text{first } K \text{ highest } M_i)$

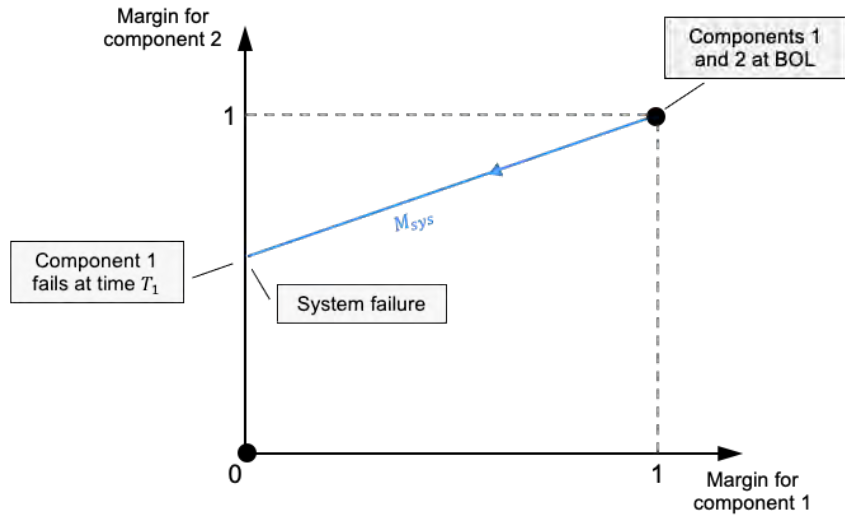


Figure 35. Evolution of M_{sys} for two components in a series configuration.

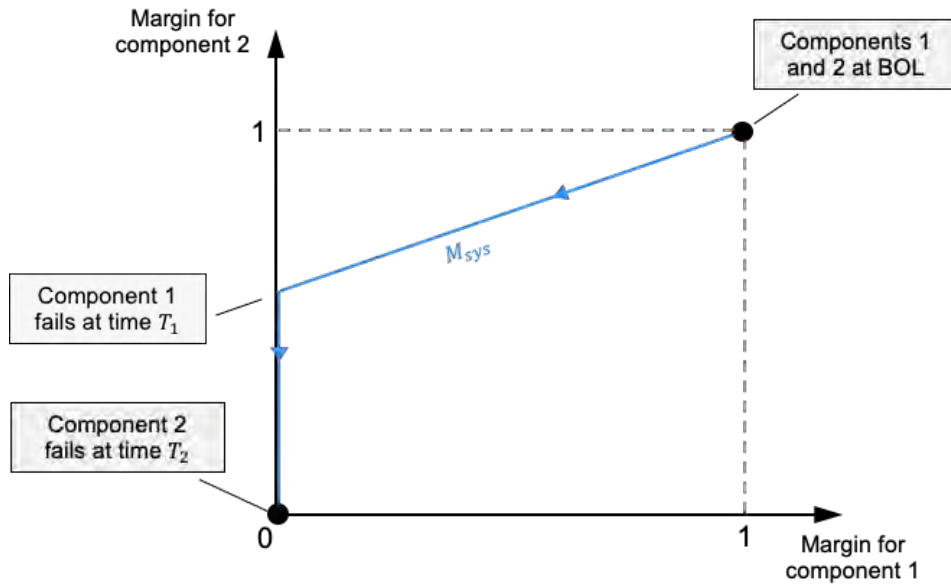


Figure 36. Evolution of M_{sys} for two components in a parallel configuration.

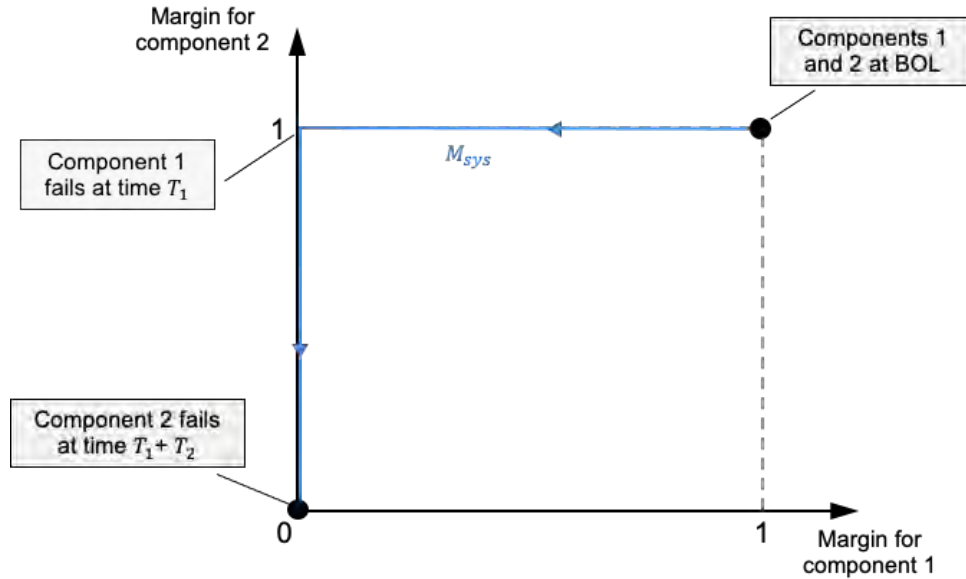


Figure 37. Evolution of M_{sys} for two components in a standby configuration.

5. DECISION-MAKING

The scope of this section is to cover the last portion of the RIAM project research area, which covers the development of decision-making tools. This area requires information generated by the other two research areas (ER data analytics and system reliability modeling) in various forms depending on the specific use case.

5.1 From Margin to Plant Resource Optimization Methods

As indicated in Section 4, a margin-based reliability modeling approach is able to quantify component health given available ER data and provide insights about the most critical components that might negatively affect system operation (through the reliability measures described in Section 4.1). The next step is to decide which maintenance operations should be performed to guarantee future system operation. This step requires an additional piece information regarding the timing aspect associated with a component failure. Such an aspect is captured by the concept of *urgency*, which we have defined as the amount of time available between now and when restoration operations need to start to avoid component failure (see Figure 38). From a practical standpoint, the component urgency estimation requires two time values:

- Estimated failure time from prognostic data (i.e., through RUL) or from margin calculations (see Section 4.4.3 and Figure 29)
- Time required to restore component health. If the restoration process requires the replacement of the component, the time value can include the time to: obtain the new component (procurement time), replace the old component, and install the new component. If the restoration process requires onsite activities (e.g., chemical treatment of corroded portions of the component), the restoration time can include the time to: take the component offline, perform the restoration activity, and take the component back online.

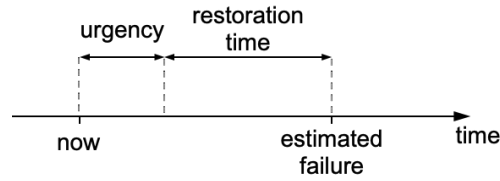


Figure 38. Graphical representation of urgency given estimated component failure and required restoration time.

At this point, we can select the most critical maintenance operations based on their urgency and the reliability importance of the components by plotting all operations on an urgency vs. importance plot as indicated in Figure 39. Depending on the industrial and decision-making context, this 2D plot can be partitioned in multiple regions where the range of each dimension is divided into two or three intervals. The selection of the operations that should be performed to guarantee future system operation can be performed by choosing the operations in selected partitions of the urgency vs. importance plot. As an example, for the case shown in Figure 39, the operations landing in the red, dark orange, and light orange sectors should be chosen.

5.2 Summary of Optimization Methods Development

As mentioned in Section 1, the RIAM project is focusing on developing methods to optimize plant resources (e.g., SSC, personnel, and ER activities). These methods can be classified as either sampling methods or optimization methods. Sampling methods are mainly designed to propagate data and model uncertainties (e.g., investment evaluation). Optimization methods are designed to determine the best solution to a problem that satisfies specified constraints that, for example, account for logic (e.g., precedence requirements) and limited resources (e.g., stay within a yearly operations and maintenance budget).

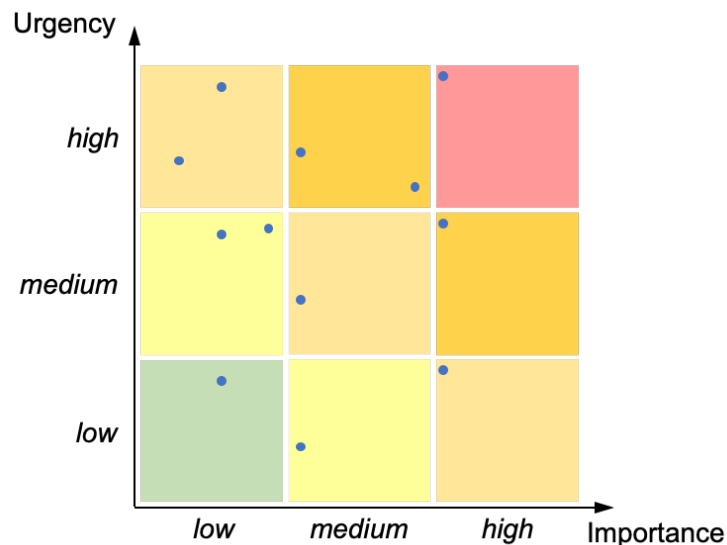


Figure 39. Selection of the most critical components in an urgency-importance diagram.

Depending on the problem to be solved and type of data available, the optimization algorithm to be used differs. As an example, the data structure might be either discrete or continuous in nature. In addition, the problem under consideration might require a specific data set or, alternatively, a specific model (balancing system reliability, availability, and cost) that changes the problem structure depending on the considered boundary conditions. Figure 40 shows in a graphical form the algorithms developed within the RIAM project for plant resources management. These methods have been classified depending on the data structure for key decisions (either discrete or continuous) and the method that the problem requires (i.e., data or model based).

The first class of methods are data-based schedule prioritization and optimization algorithms (formulated in both a deterministic and stochastic form). These methods are based on several variants of the knapsack problem (simple, multidimensional, and multiple choice). Recently, two additional versions of schedule optimization methods were implemented by the RIAM project. The first reformulates the capital budgeting problem in a distributionally robust form, which allows the user to rely on data directly rather than proposing a probability distribution from the data itself. The second version reformulates the capital budgeting by optimizing performance while explicitly accounting for risk measures that limit low-probability high-cost events.

Model-based optimization methods are designed to find the maxima or minima of a generic model (i.e., an external model), where the response of the model can change throughout the optimization process. In a mathematical form, we are dealing with models that can be considered a black box, where it is possible to define its input $\mathbf{x} = [x_1, \dots, x_N]$ and output variables $\mathbf{y} = [y_1, \dots, y_M]$. A model-based optimization method aims to minimize (or maximize) one element³ of the output variables:

$$\begin{aligned} \min_{\mathbf{x}} \quad & y_1 \\ \text{s. t.} \quad & \mathbf{x} \in \Xi \\ & \mathbf{y} = \mathbf{F}(\mathbf{x}) \\ & \mathbf{G}(\mathbf{y}) \leq 0 \end{aligned} \tag{19}$$

In our case, $\mathbf{F}(\mathbf{x})$ is a model (e.g., probabilistic risk analysis code, economic model) that determines \mathbf{y} given \mathbf{x} (i.e., $\mathbf{y} = \mathbf{F}(\mathbf{x})$). In some of our applications, \mathbf{x} and \mathbf{y} are not subject to measurement or observation errors (or, more generally, the model does not possess stochastic behavior). Note that we can introduce two possible types of constraints: explicit constraints ($\mathbf{x} \in \Xi$) that limit the variable range of the input variables and implicit constraints ($\mathbf{G}(\mathbf{y}) \leq 0$), which limit the variable range of output variables.

While the data-based methods tend to not be computationally expensive with respect to a function evaluation, the model-based methods tend to require more computational resources since the model under consideration might have to be run a large number of times, and each model evaluation might take a considerable amount of time (from minutes to hours) to execute.

In the past FYs, there were many efforts to develop and test: 1) discrete optimization methods that are part of LOGOS⁴ and 2) model-based evolutionary methods based on GAs. During FY-22, we have focused on the testing and performance evaluation of the LOGOS methods and creating of the XSD schema. This activity has been performed in collaboration with Quantum Ventura⁵ as part of the Small Business Innovation Research and Small Business Technology Transfer award last year. The XSD schema was created to provide a schema for XML documents that are actually used as input files for all LOGOS optimization methods. An XSD schema provides information about the structure of an XML file in terms

³ Note that here we are focusing on a single-objective optimization problem; another class of optimization problems involves multiple objectives, which involve more than one objective function to be optimized simultaneously. We approach this class of optimization problems using a Pareto frontier analysis (see Section 5.2).

⁴ LOGOS official GitHub repository: <https://github.com/idaholab/LOGOS>.

⁵ <https://www.quantumventura.com/>

of nodes and attribute that are allowed to be defined in the XML file itself. An XSD schema was developed for each discrete optimization method in LOGOS. This allows the developers at Quantum Ventura to create a flexible (html based) graphical user interface for the LOGOS optimization methods.

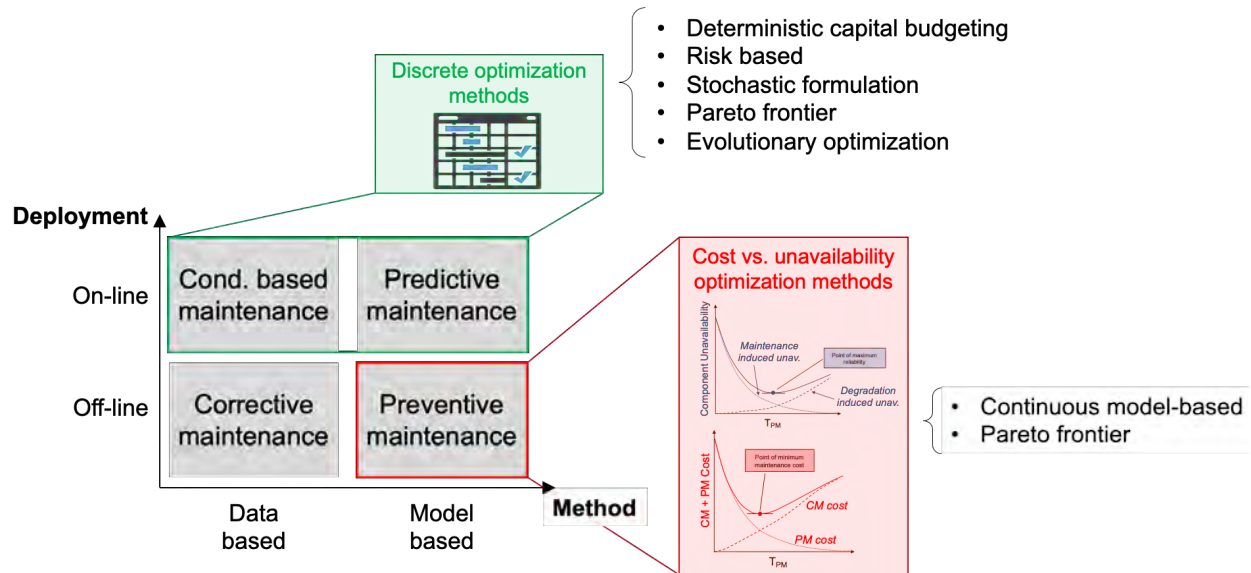


Figure 40. Classes of maintenance approaches and their corresponding optimization methods.

5.3 Testing and Development of Evolutionary Methods

GAs (Eiben and Smith 2003) represent a relevant class of evolutionary optimization methods for both continuous and discrete optimization problems. From a high-level perspective, these methods act on a population of sampled points $(x, F(x))$ (rather than focusing on a one-sample-at-a-time mindset), and they iteratively combine pairs of points to generate a new generation of points with higher quality.

The main operators employed by GAs during the optimization process are (see also Figure 41 for the main GA workflow):

- *Crossover*: the encodings of two chromosomes are mixed to generate two new encodings
- *Mutation*: the encoding of a chromosome is altered by randomly changing the value of a single element of the chromosome
- *Replacement*: the population of chromosomes is updated by removing chromosomes with low fitness or high generational age value and keeping chromosomes with high fitness or low generational age.

During FY-22, several unit tests for main GA operator methods, including the ones listed above, have been developed in order to improve the robustness of the RAVEN GA optimization class. The same tests are part of the normal regression test that needs to be validated for every change of the RAVEN source code as part of the RAVEN version control.

An existing issue about GA optimization is the ability to monitor the performance of the GA optimization method. We tackled this issue by developing a series of plots automatically generated by RAVEN at the end of the optimization process. These plots graphically represent the evolution of the input x and output variable $F(x)$. This evolution is captured every time the sample population $(x, F(x))$ undergoes

one iteration loop, as shown in Figure 41; we indicate each iteration as batch. Figure 42 shows one generated plot for a sample test case where the input space is 4-dimensional, $\mathbf{x} = [a, b, c, d]$, while the output variable is indicated as $ans = F(\mathbf{x})$. Starting with the initial population, a batch number of 0, generated using standard Monte-Carlo sampling, the GA method performs for each iteration/batch the set of operations indicated in Figure 41 on such a population. At each iteration, population input variables \mathbf{x} are computed, and the corresponding output values $F(\mathbf{x})$ are obtained from the employed model. Figure 42 plots the evolution of each iteration and batch for the considered input (i.e., $\mathbf{x} = [a, b, c, d]$) and output (i.e., ans) variables.

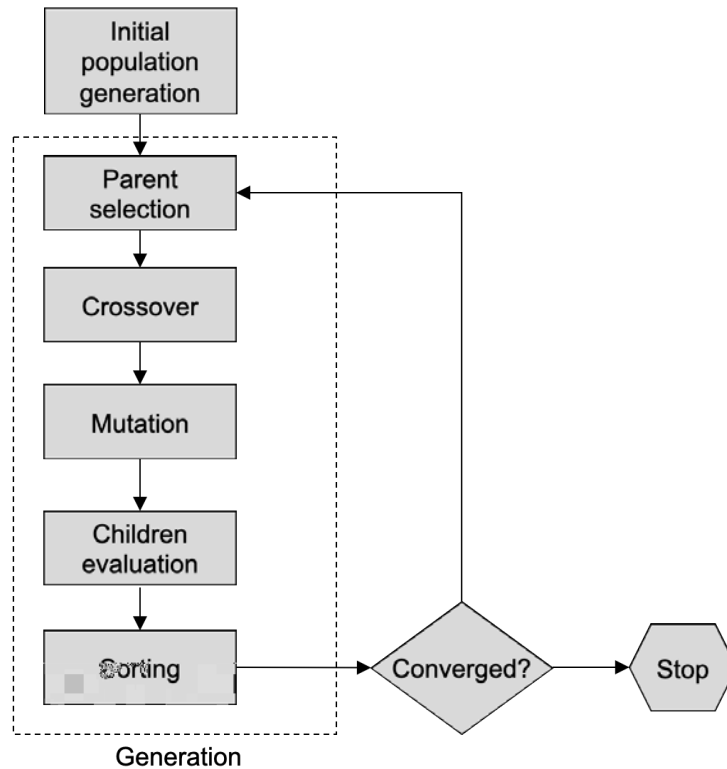


Figure 41. Graphical representation of the main GA workflow.

Another class of developed GA monitoring plots is shown in Figure 43 where the goal is to monitor the distribution of the population in the input space \mathbf{x} as GA operations shown in Figure 41 are performed for each batch. This plot captures the population distribution in a parallel coordinate plot where each line corresponds to a single chromosome and each like links the values of each dimension of \mathbf{x} for such chromosome.

Population Plot

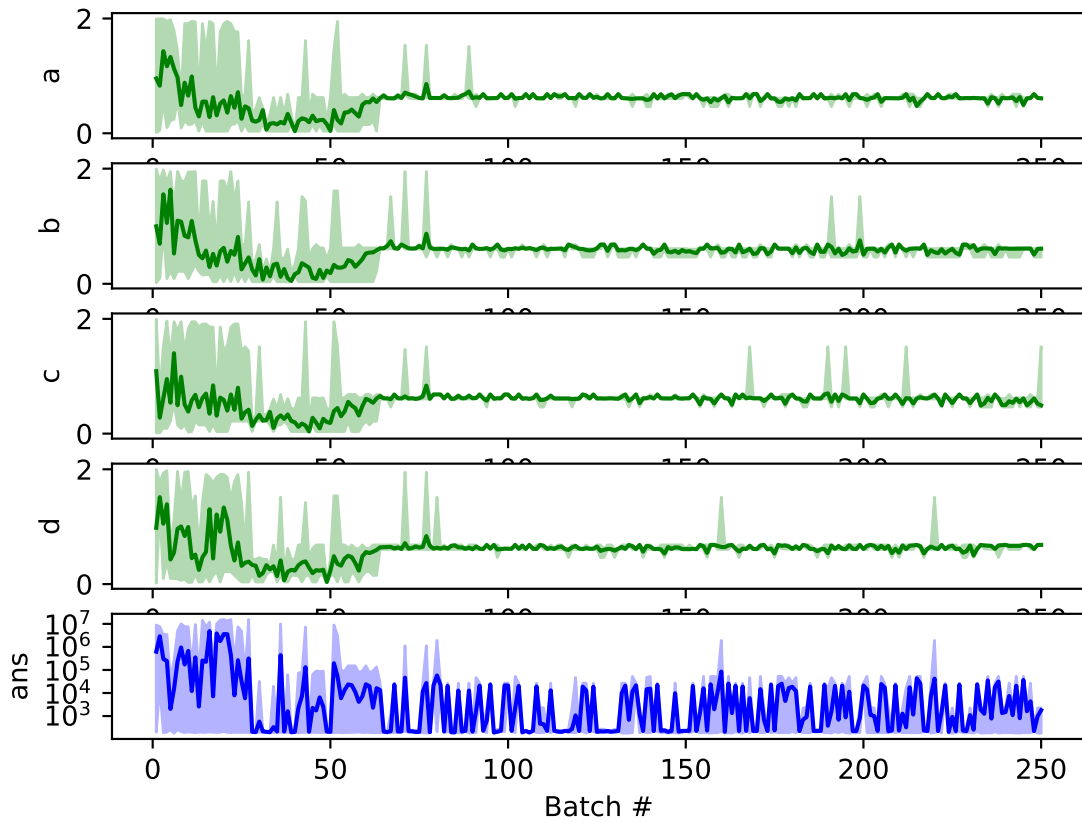


Figure 42. Plot representing the temporal evolution of the population input (i.e., $[a, b, c, d]$) and output (i.e., ans) variables.

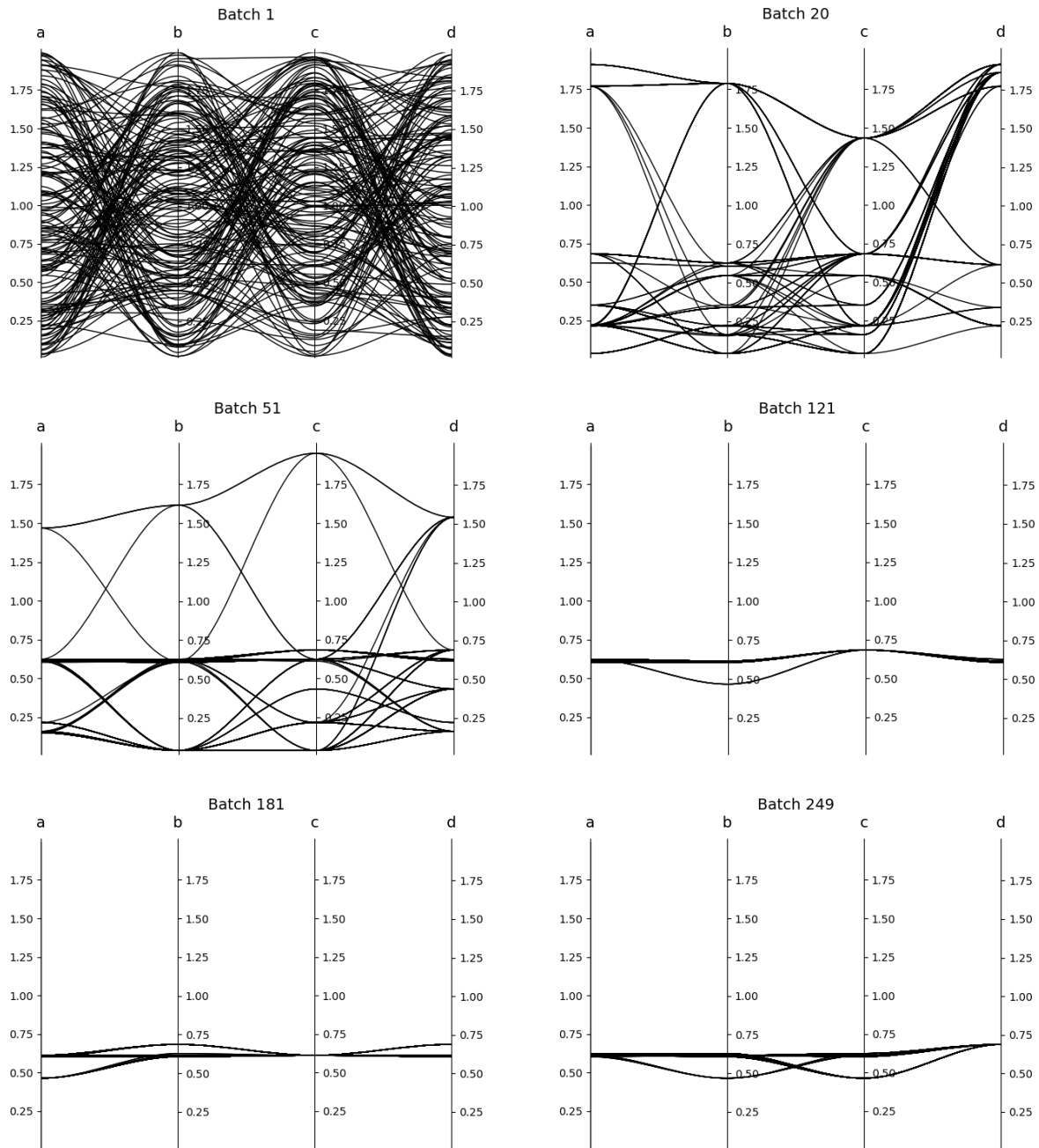


Figure 43. Parallel plot representing population values in the input space (represented by four variables ranging from a to d) for different GA iterations (indicated with batch numbers).

6. CONCLUSIONS

This report has summarized the RIAM research and development activities during FY-22. The RIAM project is creating a direct link between ER data and decision-making by focusing on three main areas: ER data analysis, reliability modeling, and plant resources optimization. In the past, the nuclear industry started addressing deficiencies in these three areas with various degrees of success. In this respect, the RIAM

project has chosen a few critical points in these three areas and developed methods and tools to overcome these critical points with innovative methods.

Regarding the analysis of ER data, we have presented a series of methods and models designed to analyze ER data with a particular focus on textual data. We have introduced an approach to extract quantitative information from ER textual data, such as IRs. Rather than focusing on ML heuristics, the system view of the SSC (through an OPM diagram) provides the knowledge required by our data analysis methods to extract knowledge from the textual data retrieved by IRs and work orders and identify possible causal links again using the SSC OPM diagram.

Regarding the system digital modeling, we have continued to develop a new way to perform reliability modeling. This margin-based method allows for a complete integration of several types of ER data (from condition to prognostic data), and it directly supports decisions that plant system engineers make regularly throughout the plant lifecycle. As indicated throughout the paper, this approach is not intended to contrast classical reliability models. On the other hand, it is designed to support a different kind of reliability question and support dynamic decisions rather than static ones (i.e., performance-related). A margin-based interpretation of reliability transforms the concept from one that focuses on the probability of occurrence to one that focuses on assessing how far away (or close) an SSC is to an unacceptable level of performance or failure. This transformation has the advantage that it provides a direct link between the SSC health evaluation process and standard plant processes used to manage plant performance (e.g., the plant maintenance and budgeting processes). The transformation also places the question into a more familiar and readily understandable form for plant system engineers and decision makers (Xingang 2021). When dealing with condition-based data (actual and archived data), we define margin as the distance between actual SSC observed conditions (e.g., oil temperature, vibration spectrum) and the condition that leads to failure (i.e., condition right before failure). Note that margin values change with time. As an example, when new component condition data that indicate component degradation are observed, the component margin decreases. Similarly, when maintenance operations are performed to such component, its health is restored and margin increases.

REFERENCES

- Alfonsi, A., C. Rabiti, D. Mandelli, J. Cogliati, C. Wang, P. W. Talbot, D. P. Maljovec and C. Smith. 2020. "RAVEN User Guide." INL/EXT-18-44465, Idaho National Laboratory.
- Baraldi, P., F. Di Maio, P. Turati, and E. Zio. 2015. "Robust Signal Reconstruction for Condition Monitoring of Industrial Components via a Modified Auto Associative Kernel Regression Method." *Mechanical Systems and Signal Processing* 60–61: 29–44. <https://doi.org/10.1016/j.ymssp.2014.09.013>.
- Borky, J. and T. Bradley. 2018. *Effective Model-Based Systems Engineering*. Springer. <https://doi.org/10.1007/978-3-319-95669-5>.
- Dori, D. and E. Crawley. 2002. *Object-Process Methodology: A Holistic Systems Paradigm*. Heidelberg: Springer. <https://doi.org/10.1007/978-3-642-56209-9>.
- Doan, S., E. W. Yang, S. S. Tilak, P. W. Li, D. S. Zisook, and M. Torii. 2019. "Extracting health-related causality from twitter messages using natural language processing." *BMC Medical Informatics and Decision Making* 19(3): 71–8. <https://doi.org/10.1186/s12911-019-0785-0>.
- Eiben, A. E. and J. E. Smith. 2003. *Introduction to Evolutionary Computing (2nd edition)*. Heidelberg: Springer. <https://doi.org/10.1007/978-3-662-44874-8>.
- Friedenthal, S., A. Moore, R. Steiner, 2008. *A Practical Guide to SysML: The Systems Modeling Language*. Morgan Kaufmann. <https://doi.org/10.1016/C2013-0-14457-1>.

- Honnibal, M., Johnson, M. 2014. “Joint Incremental Disfluency Detection and Dependency Parsing.” *Transactions of the Association for Computational Linguistics* 2: 131–142. https://doi.org/10.1162/tacl_a_00171.
- Lane, H., Hapke, H., and Howard, C. 2019. *Natural Language Processing in Action: Understanding, Analyzing, and Generating Text with Python*. Shelter Island, New York: Manning Publications.
- Lee, J. C. and McCormick, N. J. 2011. *Risk and Safety Analysis of Nuclear Systems*. John Wiley & Sons, Inc. <https://www.doi.org/10.1002/9781118043462>.
- Luo, Y., Zhang W., Fan Y., Han, Y., Li, W., and Acheaw E. 2021. “Analysis of Vibration Characteristics of Centrifugal Pump Mechanical Seal under Wear and Damage Degree.” *Vibration and Control of Fluid Machinery and Systems* 2021. <https://doi.org/10.1155/2021/6670741>.
- Mandelli, D., C. Wang, J. Cogliati, C. Smith, S. Hess, R. Sugrue, C. Pope, J. Miller, S. Ercanbrack, D. Cole, and J. Yurko. 2020. “Integration of Data Analytics with Plant System Health Program. Idaho National Laboratory Technical Report.” INL/EXT-20-59928, Idaho National Laboratory. <https://www.osti.gov/servlets/purl/1691467>.
- Mandelli, D., C. Wang, and S. Hess. 2021. “On The Language of Reliability: A System Engineer Perspective.” *Proceedings of 2021 Probabilistic Safety Assessment (PSA) Conference, Virtual, November 7–12, 2021*.
- Moerchen, F. 2007. “Unsupervised Pattern Mining from Symbolic Temporal Data.” *ACM SIGKDD Explorations Newsletter* 9(1): 41–55. <https://doi.org/10.1145/1294301.1294302>.
- Mohri, M., A. Rostamizadeh, and A. Talwalkar. 2012. *Foundations of Machine Learning*. The MIT Press.
- Nassif, A. B., M. A. Talib, Q. Nasir, and F. M. Dakalbab. 2021. “Machine Learning for Anomaly Detection: A Systematic Review.” *IEEE Access* 9: 78658–78700. <https://doi.org/10.1109/ACCESS.2021.3083060>.
- Pearl, J. 2009. “Causal Inference in Statistics: An Overview.” *Statistics Surveys* 3: 96–146. <https://doi.org/10.1214/09-SS057>.
- Rausand, M., A. Barros, and A. Hoyland. 2020. *System Reliability Theory: Models, Statistical Methods, and Applications*. John Wiley & Sons, Inc. <https://www.doi.org/10.1002/9781119373940>.
- Sadvilkar, N. and M. Neumann. 2020. “PySBD: Pragmatic Sentence Boundary Disambiguation.” In *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS)*, Virtual, November 2020, 110–114. <http://dx.doi.org/10.18653/v1/2020.nlpss-1.15>.
- Stone, G. C., I. Culbert, E. A. Boulter, and H. Dhirani. 2014. *Electrical Insulation for Rotating Machines: Design, Evaluation, Aging, Testing and Repair*. Second Edition, IEEE Press. <https://www.doi.org/10.1002/9781118886663>.
- Swales, J. M. and C. Feak. 2012. *Academic Writing for Graduate Students: Essential Tasks and Skills (3rd edition)*. Ann Arbor, MI: University of Michigan Press. <https://doi.org/10.3998/mpub.2173936>.
- William, S. 2004. *The Object Primer: Agile Model Driven Development with UML 2.0*. Cambridge, UK: Cambridge University Press. <https://doi.org/10.1017/CBO9780511584077>.
- Xingang, Z., J. Kim, K. Warns, X. Wang, P. Ramuhalli, S. Cetiner, H. G. Kang, and M. Golay. 2021. “Prognostics and Health Management in Nuclear Power Plants: An Updated Method-Centric Review with Special Focus on Data-Driven Methods.” *Frontiers in Energy Research* 9: 696785. <https://doi.org/10.3389/fenrg.2021.696785>.

Young, T., H. Devamanyu, S. Poria, and E. Cambria. 2018. "Recent Trends in Deep Learning Based Natural Language Processing." *IEEE Computational Intelligence Magazine* 13(3): 55–75. <https://doi.org/10.1109/MCI.2018.2840738>.

Youngblood, R.W. 2001. "Risk significance and safety significance." *Reliability Engineering & System Safety* 73(2): 121–136. [https://doi.org/10.1016/S0951-8320\(01\)00056-4](https://doi.org/10.1016/S0951-8320(01)00056-4).

APPENDIX A: CLASSICAL RELIABILITY MODELING

The goal of classical reliability theory is to analyze the reliability properties of a system composed of several components. This is performed by propagating reliability information, in terms of the probability of failure, from the component level to the system level. In more detail, the objective is to determine the system's ability to perform its design function(s) given the failure probability of each component being part of the system itself. Component reliability directly measures the ability of a component to perform its function and is defined over a time interval. Mathematically speaking, component reliability $R(t)$ can be translated as the probability that the system will fail past the specified time interval, as follows:

$$R(t) = Pr(t' > t) = \int_t^{\infty} f(t') dt' \quad (\text{A.1})$$

where $f(t')$ indicates the failure probability distribution function.

Typically, $f(t')$ is expressed using an exponential model:

$$f(t') dt' = \lambda dt' e^{-\lambda t'} \quad (\text{A.2})$$

which indicates the probability that a component failure occurs in the $[t', t' + dt']$ (i.e., the term $\lambda dt'$) given that the component survived up to t' (i.e., the term $e^{-\lambda t'}$). The term λ is a failure rate and can be used to determine the MTTF as:

$$MTTF = \frac{1}{\lambda} \quad (\text{A.3})$$

As indicated above, the goal is to assess system reliability; to accomplish this task, we need information about the reliability of its components and the architecture of the system itself. System architecture captures dependencies and redundancies between components. Typically, the system architecture is modeled using block diagrams where the basic configurations are:

- Series configuration (see Figure 44, top left): blocks in a series configuration are all required to operate
- Parallel configuration (see Figure 44, top right): blocks in a parallel configuration provide redundant operation
- Standby configuration (see Figure 44, bottom left): a block initially provides the desired function while the second block provides the function when the first one fails
- KooN configuration (see Figure 44, bottom right): a special case of parallel configuration where out of N parallel (i.e., redundant) components, only K (where $K < N$) are required to successfully operate.

Table 33 provides the basic equations to determine system reliability and corresponding MTTF for the four configurations we considered (see Figure 44). These equations will be the basis to compare classical and margin-based reliability calculations (see Section 4.8).

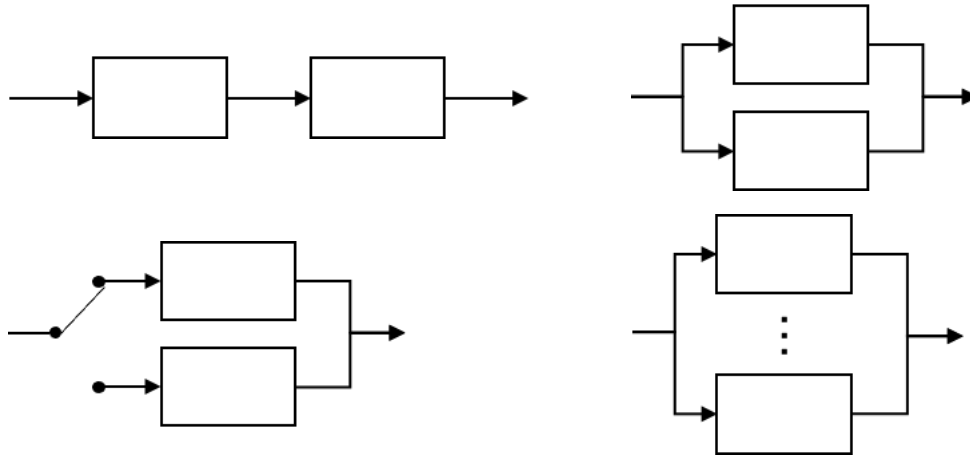


Figure 44. Graphical representation of the most common block configurations: series (top left), parallel (top right), standby (bottom left), and KooN (bottom right).

Table 33. Summary of classical reliability calculations for the considered four configurations.

Configuration	Configuration Reliability	Configuration MTTF
Series	$R(t) = \prod_{i=1}^N R_i(t)$	$MTTF = \frac{1}{\sum_{i=1}^N \lambda}$
Parallel	$R(t) = 1 - \prod_{i=1}^N (1 - R_i(t))$	$MTTF = \frac{1}{\lambda} \sum_{i=1}^N \binom{N}{i} \frac{(-1)^{i+1}}{i}$
Standby	$R(t) = R_1(t) + \lambda t R_2(t)$	$MTTF = \frac{1}{\lambda} + \frac{1}{\lambda} = \frac{2}{\lambda}$
KooN	$R(t) = \sum_{i=k}^N \binom{N}{i} R(t)^i (1 - R(t))^{N-i}$	$MTTF = \sum_{i=k}^N \binom{N}{i} \int_0^{\infty} R(t)^i (1 - R(t))^{N-i}$

APPENDIX B: CUT SETS AND PATH SETS

Common reliability analysis methods are designed to work in the “failure space” where the goal is to identify the combination of events that yield adverse consequences. These combinations of events are typically represented the MCSs as the logic product of BEs (e.g., failure of components or failure to perform recovery actions).

However, from a system engineer perspective we are interested to work in “success space” where the objective is to identify combination of events that guarantee system operation. These new combinations of events are represented by the MPSs of the system under consideration. In formulating a safety case for a facility, it is beneficial to do a “prevention analysis” in order to identify the combinations of success paths that (together) provide needed levels of functional reliability, required redundancy, and required diversity. In addition, if something goes wrong, and we need to compensate for it lest we lose the function, it is useful to understand what success paths remain available and which of them are more reliable than the others.

Given a defined system architecture, MCSs and MPSs are complementary objects: while MCSs are generated from Boolean structures, such as FTs, where the top event is defined as a “system failure,” MPSs are generated from the FT having top event defined as “NOT system failure.” In addition, while each MCS guarantees system to fail, each MPS guarantees system to operate successfully. In mathematical terms, this can be represented as:

$$\text{System failure} = OR(MCS_1, \dots, MCS_N) \tag{B.1}$$

$$\text{System success} = OR(MPS_1, \dots, MPS_M) \tag{B.2}$$

As an example, let’s consider the system shown in Figure 31 composed of seven components, A–G. The determination of the MCSs corresponding to the system of Figure 31 can be performed by generating the corresponding FT:

$$\begin{aligned} \text{System failure} &= A (B OR (C OR D) (E F OR G)) = \\ &= AB + ACEF + ACG + ADEF + ADG \end{aligned} \tag{B.3}$$

The set of MPS can be obtained by solving: $\neg(\text{System failure}) = NOT(\text{System failure})$. The lists of corresponding MPSs and MPSs is indicated in Table 34.

Table 34. Lists of MCSs and MPS for the system shown in Figure 31.

#	MCS	#	MPS
1	AB	1	/A
2	ADG	2	/B /C /D
3	ACG	3	/B /E /G
4	ADEF	4	/B /F /G
5	ACEF		

APPENDIX C: OPM MODELING

This appendix is presenting the basic elements found in OPM diagrams. In this respect, these elements are listed in Table 35 along with their semantic description. These elements are widely used in the RIAM project, as indicated in the OPM diagrams of the MFW system (see Appendix D). Table 36 provides basic OPM diagrams that link objects with processes. Note that Table 36 also provides a description of the causal nature of the each OPM diagram. This causal interpretation is the basis for the causal reasoning process shown in Section 2.2 when dealing with ER data.

Table 35. Basic elements of OPM (adapted from Dori and Crawley [2002]).


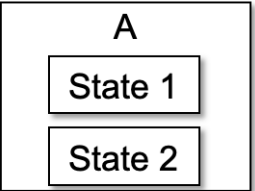
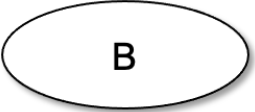


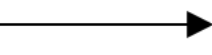
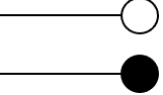
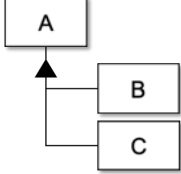


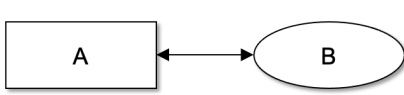
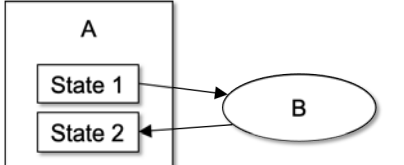


OPM Elements	Name	Description
	Object	Object A is a tangible entity that exists
	Object with states	Object A has two states: State 1 and 2
	Process	Process B transforms an object
	Aggregation link	Link designed to decompose an object into its basic elements
	Exhibition link	Link designed to define attributes of an object
	Procedural transforming link	Link designed to indicate a transportation activity between a process and an object
	Procedural enabling links	Link designed to represent objects that support a process

Table 36. Causal relations from OPM diagrams.

OPM Diagram	Description	Causal Relation
	Object A is composed by Objects B and C	An abnormal behavior of A can be caused by either B or C
	Process B yields Object A	An abnormal status of A can be caused by B
	Process B consumes Object A	An abnormal behavior of B can be caused by A
	Process B affects Object A	An abnormal behavior of B can be caused by A. An abnormal status of A can be caused by B
	B is changing state of A from State 1 to State 2	An abnormal transition from State 1 to State 2 of A can be caused by B
	Process B requires Object A	An abnormal behavior of B can be caused by A
	Object A handles Process B	An abnormal behavior of B can be caused by A

APPENDIX D: MFW OPM MODELS

This appendix contains the OPM models of several components of a pressurized-water reactor (PWR) MFW system, as shown in Figure 45:

- General schematics of a MFW PWR system (see Figure 46)
- Centrifugal pump (see Figure 4)
- Turbine-driven pump (see Figure 47)
- Motor-operated valve (see Figure 48)
- Shaft bearings (see Figure 49)
- Feedwater heater (see Figure 50)
- Deaerator (see Figure 51)

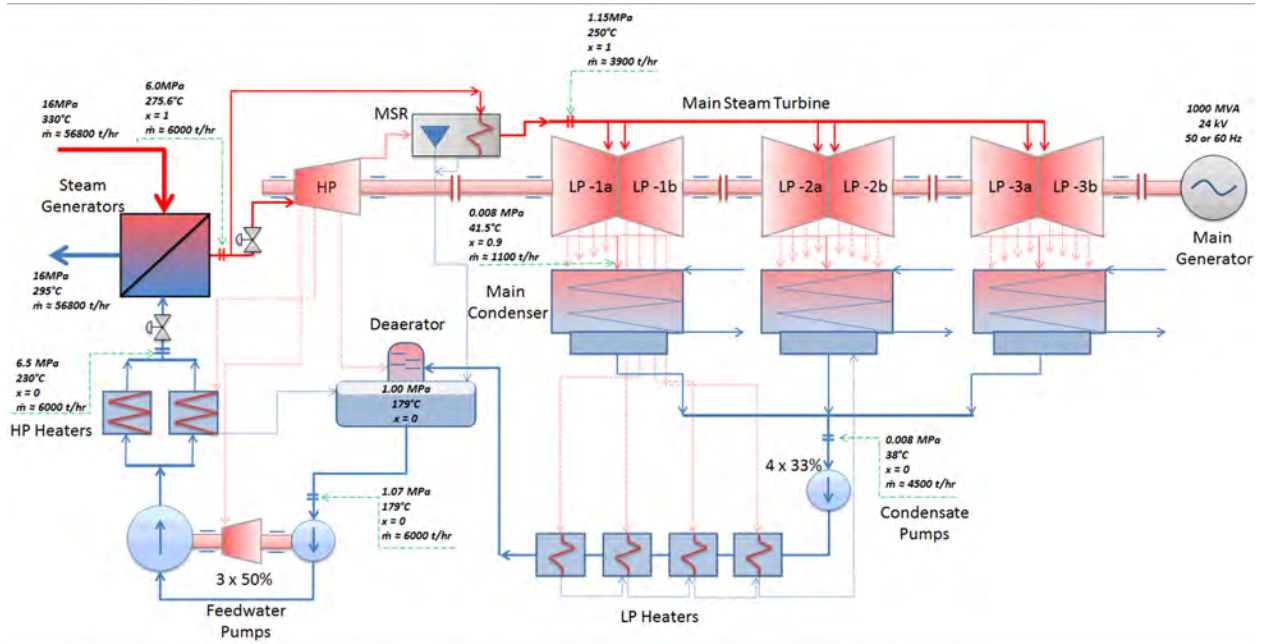


Figure 45. Schematics of a PWR MFW system (source: www.nuclear-power.com).

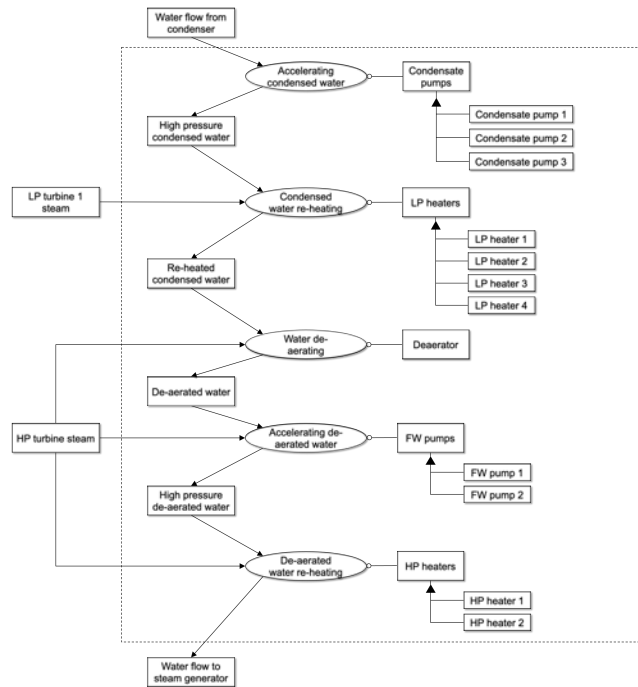


Figure 46. OPM model of MFW system.

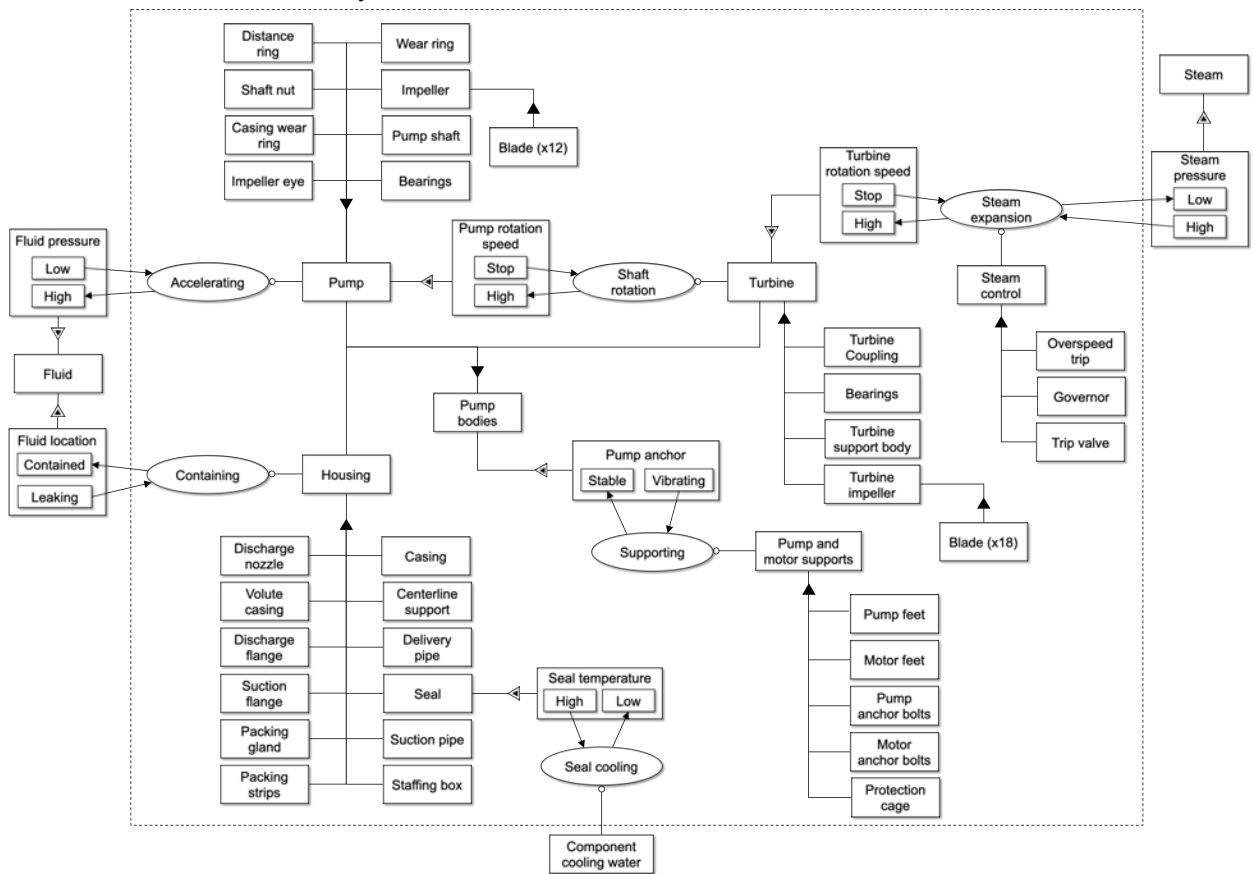


Figure 47. OPM model of a turbine-driven pump.

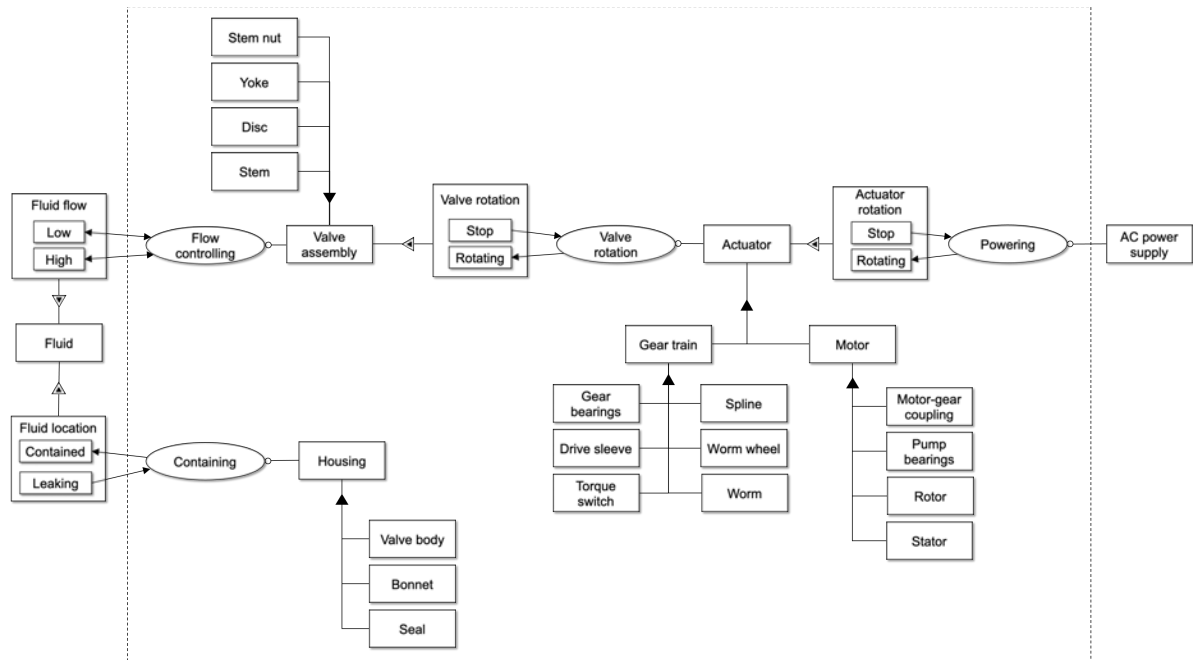


Figure 48. OPM model of a motor-operated valve.

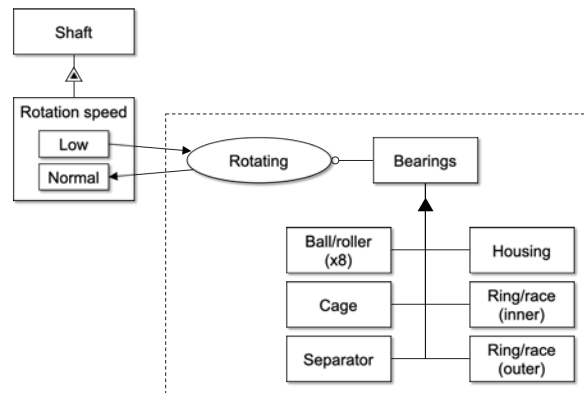


Figure 49. OPM model of a shaft bearings.

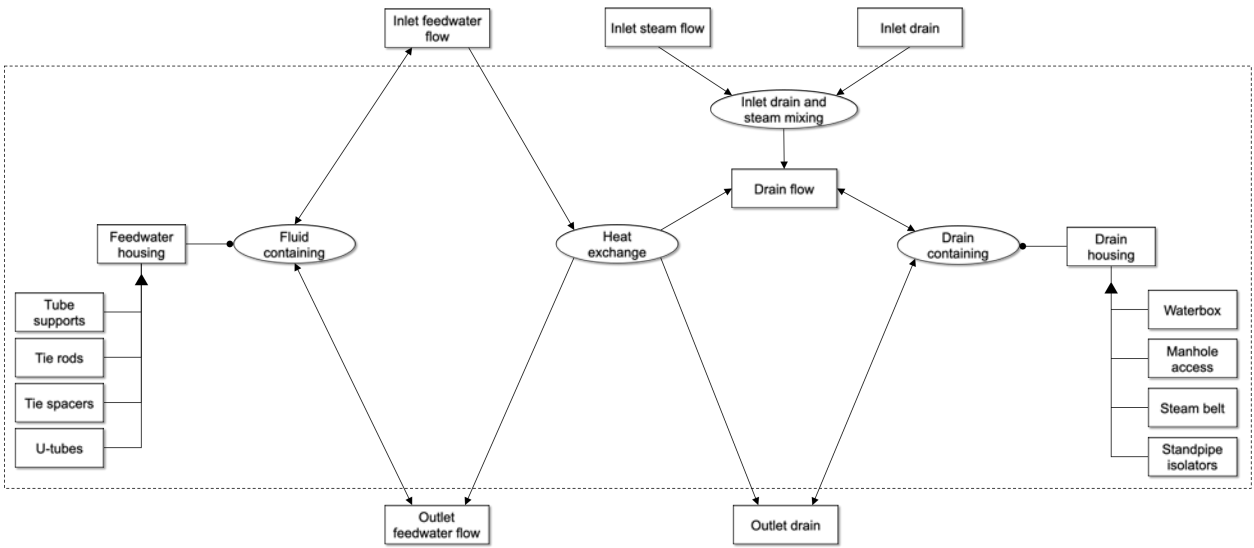


Figure 50. OPM model of a feedwater heater.

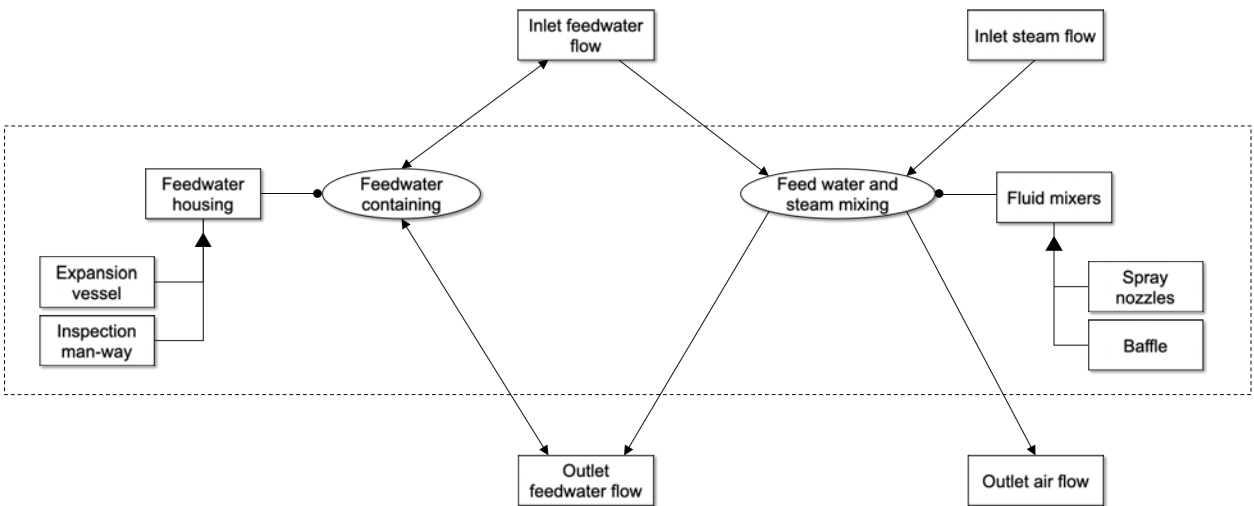


Figure 51. OPM model of a deaerator.

APPENDIX E: MFW SYSTEM RELIABILITY DATA

Feedwater pumps

- a. Outages due to oil system problems: 41% of the cases
 - Hydraulic control oil system: oil pump failure, dirty oil, governor, control, steam flow/feed mismatch, and low steam generator level trip. Outages occurred when contaminants (paint, rust, dirt, etc.) entered the control system and blocked small control orifices or blocked the seats of ball check valves causing improper closure. These blockages cause the speed controller to overspeed due to increased control pressure when control is transferred from the governor valve positioner to the governor. The outages are thought to be caused by the present design in which the hydraulic control oil and lubrication oil use the same reservoir.
 - Oil line leaks: caused by wear and aging of connecting hoses.
 - Miscellaneous Turbine Driven Feedwater Pump Oil Problems: include leaking gaskets and valves. These outages are caused by wear and aging during normal operation.
- b. Outages due to Bearing Problems: 10% of this type were prevalent in 20% of the stations. Vibration, and repair or replacement of bearings usually attributable to bearing wear. The cause of bearing wear or degradation is normally a lack of sufficient lubrication or particles in the lube oil or bearing. Either of the above will cause accelerated wear of bearings.
 - Repair or Replacement: No specific reasons for the repairs or replacement were reported. Bearing wear and vibration are the usual causes.
 - Outages due to Vibration: The causes stated were improper clearance, wear, misalignment, unbalance and vibration.
- c. Outages Due to Steam Leaks: The cause is attributed to the wear of gaskets due to normal expansion and contraction of joints.
- d. Outages Due to the Pump Driver:
 - Outages due to the turbine driver: The cause was damage to the blades resulting in vibration.
 - Outages Due to the Motor Driver: The cause was due to electrical problems
- e. Outages Due to Impeller Failure: 4% of the cases. Causes include foreign objects lodged in the impeller. These foreign objects were pieces of equipment from failed parts of upstream equipment.
- f. Outages Due to Casing Failure: 3%. Due to porous welds that contained slag inclusions and sand and gas cavities in the castings. The cause of these failures was attributed to poor castings and loose specification.
- g. Outages Due to Miscellaneous Reasons: 6%, include faulty valves, cracked couplings, shaft failure, etc.
- h. Outages Without Identifiable Causes: 19%, not enough information.

Feedwater valves. Many Steam Generator Level trips at low powers and at other power levels were reported but with unreported causes. Failure to control these valves (manual control) is considered the predominant of causes. Automatic control at low power would thus help the operator for better start up and low power operation. The control valve trim should be matched to the type of feedpump. In addition, the valve stroke has a definite impact on level control and stability. The valve stroke should be as long as possible (approximately 90%) over the operating range of 0 to 100% power. The longer stroke allows better control, i.e., more stability, at lower power levels.

Main condenser. Condenser tube leaks. Causes are tube vibration, jet impingement, corrosion/erosion and impingement plate failure. Vibration has been handled by "staking" the tubes or lacing them together. Jet impingement has been dealt with by the replacement of impingement plates; impingement plate failure has been reduced by insuring better workmanship and adding stronger supports. Corrosion/erosion has been

dealt with by changing materials mainly from admiralty to Cu-Ni combinations, stainless steel, or titanium. However, condenser tube leaks continue to occur and cause more down time each year.

Many solids and organic material are transported from the cooling water source into the condenser water box and tubes. This results in tube blockage and corrosion due to inadequate tube water flow. This in turn results in tube leakage due to corrosion and thermal shocking, especially when the turbine is opened up for maintenance. These corrosion products eventually end up in the condenser hotwell and are transported into the feedwater cycle unless adequately cleaned prior to startup. Feedwater cycle problems and steam generator problems are in part attributable to this crud transport.

A condenser tube leak means a drop in boiler pH. For many units, control of the boiler pH is the primary determinant of whether or not the contamination is severe enough to take the unit off line. Obviously, if the boiler pH cannot be controlled, and particularly if the pH drops below 8, the unit must come off line immediately.

Steam passing through the turbine (preferably sampled at the reheat steam sample station after attemperation) should contain less than 2 ppb of sodium and have a cation conductivity of less than 0.2 microsiemens/cm. Some utilities have chosen to monitor for chloride in the boiler water directly.

Feedwater heaters. Leaky feedwater heater tubes were attributed to 0.19 outage events per plant year and 0.04 percent average annual plant unavailability for the 192.4 years of PWR plant experience. Feedwater heater tube leaks occur either at the tube-to-tubesheet joints or due to failure of the tube itself, similar to main condenser problems. Degradation of joints may occur due to corrosion, erosion, tube vibration, thermal shock, chemistry changes or improper operation.

- a. Tube-to-Tubesheet Joint Leakage: Fabrication issue.
- b. Tube Vibration: causes deterioration of tube-to-tubesheet joints, mechanical abrasion of tubes rubbing against baffle and support plates, and tube fatigue. Vibration may be due to various design considerations (high velocities, inadequate tube supports, pressure fluctuations, localized high velocity steam jets, etc.), unanticipated transients or improper operation such as exceeding design parameters. Resonant vibration is extremely detrimental to tube failure.
- c. Thermal shock: Tubes and tube-to-tubesheet joints can be overstressed if subjected to large or frequent thermal shocks.
- d. Corrosion and Erosion: Contributing factors include tube entrance effects and local high velocities from manufacturing defects or deformations within the tubes, the presence of foreign materials and improper operation resulting from chemistry changes.
- e. Material degradation

Turbine

- a. Blade failures: 1.6% average unavailability contribution per plant year, 0.04 scrams per plant year and 0.10 shutdowns per plant year at fifteen of forty plants. The average shutdown duration from blade problems was 1343.3 hours (56 days).

Most failures occur in the low pressure turbine. The root causes of these blade failures were: erosion due to moisture produced in the turbine, inadequate Moisture Separator Reheater (MSR) performance, reliability feedwater system corrosion products which are carried over, corrosion from improper feedwater chemistry control, underfrequency operation and overstress. Turbine blade erosion is a function of water droplet size, steam quality, steam density, rotating blade speed, and inherent turbine and blade design features. If the variables can be controlled to an acceptable level, blade erosion can be limited. Mineral deposits on the blades can be limited by feedwater chemistry control and functional MSRs.

Improvements which can reduce or eliminate low pressure turbine blade failures are:

- Maintain feedwater chemistry within specification and provide for 100% demineralization of the condensate when required.

- Improve MSR performance and reliability.
 - Limit low frequency operation.
 - Provide temperature, pressure and vibration monitoring of turbine and MSR for load equalization among LP turbines and turbine stages.
 - Limit turbine and main steam valve testing.
- b. Bearing failures: Turbine bearing failures rank next in the total outage. Causes are: dirty oil, blocked oil passage, emulsification, improper cooling of lube oil, bearing vibration. These problems can be minimized by increasing the frequency of the oil system to identify the onset of dirt or emulsification build up, and by routine inspection of temperature and pressure gauges in the oil system. Long term solutions include the use of larger orifices in the lubrication system and improved lube oil filtration. Early detection through vibration and acoustic monitoring techniques can enhance the likelihood of corrective action prior to bearing failure.
 - c. Steam leaks: root causes include the following: erosion, impingement, uneven heating, and improper joint and gasket sealing. Minor steam leaks (early stages of large leaks) can be temporarily repaired using a sealant or patch and minimize the number of non-welded joints.
 - d. Vibration and Imbalance: Causes of turbine vibration and imbalance are: shaft and rotor distortion, uneven blade erosion, mineral deposits on blades, improper clearance tolerances, misalignment, internal mechanical looseness, foundation and case distortion, internal rubbing, uneven bearing wear and a bowed rotor.
- b. Degradation causes imbalance, which leads to vibrations. Vibration and acoustic monitoring programs could enhance the capability of detecting premature secondary problems and their locations. Temperature and pressure diagnostic monitoring of the steam systems and turbine could also aid in root cause determination and corrective action prior to mature problems. Proximity detectors could also monitor turbine alignment.

Moisture Separator Reheater (MSR). Over seventy percent of the outages were caused by tube leakage. The remaining outages were caused by problems with baffle plates, demisters, separators, handholes and high water level trips. The root causes of tube leakage are similar to other heat exchanger problems, thermal fatigue, vibration, corrosion and erosions. Factors which contribute to thermal fatigue, vibration erosion and corrosion are:

- crud deposition on interior and exterior reheater tube surfaces as well as separator surfaces
- inadequate tube bundle drainage
- condensation in tubes
- condensate subcooling of tubes
- temperature gradients between tubes and tubesheet
- non-uniform steam flow
- control and intercept valve testing
- vibration overstress
- use of materials not compatible with the service
- feedwater chemistry control problems

Turbine Control, Stop and Intercept Valves

- a. Valve Malfunctions: The problems include valves not opening, valves not closing, valves sticking, valve failures, and spurious valve closures
- b. Valve Repair or Replacement: due to leaks and include some stem and disc replacements. Causes can be attributed to valve wear, some beyond repair.
- c. Valve Modifications and Miscellaneous Reasons: modifications required by the manufacturer, and due to auxiliary equipment such as servomotor and relay failure.

Main Steam Isolation Valves. Nearly 65 percent of the significant MSIV outage events referenced in the data base resulted as automatic scrams. Auto scrams are caused by inadvertent closures which lead to steam

generator high pressure trips. Auto scrams account for only one-half of the overall MSIV related unavailability hours, thus the data supports the premise that inadvertent closures rather than repairs are the most important aspect of MSIV unavailability to address.

Component supports. hydraulic shock suppressors, more commonly referred to as snubbers. The basic cause for the incidents is the loss of hydraulic fluid which leads to a loss of function of safety related snubbers. The loss of hydraulic fluid can be directly attributed to seal failures or potential seal failure on 100% of the recorded incidents. The causes of the seal failures are:

- a. Broken accumulator springs which have caused seal surface scoring and subsequent leak paths (46.6% of incidents).
- b. Seal material incompatibility (26.6%) which lead to adverse reactions to silicon hydraulic fluids leading to seal degradation and leakage.
- c. Improper installation of snubbers during which seals were damaged (26.8%) leading to the remaining seal leakage problems.