SAFETY INSTRUMENTED SYSTEM FOR PROCESS OPERATION BASED ON REAL-TIME MONITORING

CEN KELVIN NAN

SAFETY INSTRUMENTED SYSTEM FOR PROCESS OPERATION BASED ON REAL-TIME MONITORING

by

Cen Kelvin Nan, B.Eng

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Engineering

Faculty of Engineering and Applied Science

Memorial University

April, 2007

St. Johns Newfoundland

Your file Votre référence
ISBN: 978-0-494-31272-8
Our file Notre référence
ISBN: 978-0-494-31272-8

ABSTRACT

Many industrial processes pose many potential threats to life or environment, especially in case of failure. Hazardous, flammable and reactive materials are often processed at elevated temperatures and pressures. The hazards posed by those materials need to be controlled and managed in order to improve the process safety. Safety Instrumented System (SIS) is a widely recognized tool by a number of industry sectors to prevent those hazards and thus reach the required safety objective. Recently, the process industry has started to realize the significance of SIS.

Due to the increased process complexity and possible instability in operating conditions, the existing control systems have limited ability to provide practical assistance to both operators and engineers. Therefore, much attention has been focus on suitable designs of system control components. This thesis proposes a new methodology for fault diagnosis, safety function formulation, and to implement safety instrumented system based on real-time monitoring. The methodology is comprised of three stages. The first stage is to model and simulate the target process system according to the observed system behaviors. The second stage is to adopt knowledge-based fault diagnosis technique, which implements the valuable knowledge from the experts and operators as well as a vast databank of information from a variety of sensors, for making optimal decision regarding current state of the process operation. Fuzzy logic is also used in this stage to make inferences based on acquired information (real-time data) and the knowledge. This stage is a fundamental part of the proposed methodology.

A computer-aided tool, implementing previous two stages, is developed on the platform of G2 expert system platform using GDA (*G2 Diagnostic Assistant*) components in the third stage. This tool is subsequently used to verify the methodology performance through both industrial and simulated data.

The proposed methodology is straightforward, flexible and easy to understand. Moreover, the developed fault diagnosis safety function may be utilized in developing various safety instrumented systems.

ACKNOWLEDGMENTS

The author wishes to express sincere appreciation to the Faculty of Engineering and Applied Science for affording him the opportunity of conducting this work. Particularly, the suggestions and guidance of the author's supervisors, Dr. Faisal Khan and Dr. M. Tariq Iqbal, are greatly appreciated.

Thanks are also owed to author's colleagues from IIC labs and friends, Viren Panchal and Ying Zhou from Dalhousie University, for their help and assistance on ARSST device.

The work could not have been completed without the financial support from Natural Science and Engineering Research Council of Canada (NSERC) and AIF Inco Project grant to Dr. Faisal Khan and Dr. M. Tariq Iqbal.

The patience and understanding of the author's parents and family are also appreciated.

TABLE OF CONTENTS

LIST OF TABLES

Number *Page*

LIST OF FIGURES

Number *Page*

LIST OF ABBREVIATIONS

AICE	American Institute of Chemical Engineering
ALARP	As Low As Reasonably Practicable
BPCS	Basic Process Control System
DAS	Data Acquisition
DCS	Distributed Control System
ESD	Emergency Shut Down
FDD	First Discrete Derivative
FIS	Fuzzy Inference System
FTA	Fault Tree Analysis
GDA	G2 Diagnostic Assistant
GSI	G2 gateway Standard Interface
HAZOP	Hazard and Operability
IEC	International Electrotechnical Commission
IFD	Information Flow Diagram
IPL	Independent Protection Layer
LOPA	Layer of Protection Analysis
MF	Membership Function
PFD	Probability of Failures on Demand
PSV	Pressure Safety Valve
QRA	Quantitative Risk Analysis

ROC	Rate of Change
RRF	Risk Reduction Factor
SDD	Second Discrete Derivative
SF	Safety Function
SI	Similarity Index
SIL	Safety Integrity Level
SIS	Safety Instrumented System
TMR	Triple Modular Redundant

LIST OF SYMBOLS

f_i^C	Frequency for consequence C for initiating event i
f_i^I	Initial frequency for initiating event i
Fp	Protected Risk Frequency
Fnp	Unprotected Risk Frequency
kPa	kilopascal
PFD^{ij}	Probability of failure on demand of jth IPL that protects against consequence C for initiating event i.
Psi	Pound-force per square inch
$S_{P_i P_i^*}$	Similarity between primitive P_i and P_i^*
Tr	(Process)Trend
ω_n	Undamped natural frequency
ξ	Damping coefficient
τ	Time constants

Chapter 1

INTRODUCTION

1.1 Safety Instrumented System (SIS)

The chemical industry has made great strides over last 20 years toward improving process unit performance and safety operation (Summers, 2006). This improvement has been achieved through a variety of techniques, which are aided to identify and manage risks. Although each country has different programs and standards for implementing this improvement, the concept of process safety management is well known. Over last 10 years, process industry has made significant investment in research, resources, and system upgrades, minimizing risks and hazards.

System safety and reliability are the main parameters to ensure system design, development, and operation for a process facility. There are usually high demands on the safety performance of systems where the consequences of accidents are large. Safety Instrumented System (**SIS**), a system independent of Basic Process Control System (**BPCS**), is designed to take action to maintain the process safety in the event of malfunction. The International Electrotechnical Commission (**IEC**) 61508 (2000) standard defines SIS as "*a system composed of sensors, logic solvers and final-control elements for the purpose of taking the process to a safe state, when predetermined conditions are violated*" .

In general SIS aims to reduce risk to an acceptable or As Low As Reasonably Practicable (**ALARP**) level using risk-based approach. The greater the process risk, the more effective the SIS must be in order to control the risk (Cusimano and DiNapoli, 2001). The ALARP principle provides a general objective of SIS, which is to reduce the frequency at which a hazard may occur to an acceptable or at least a tolerable level. It should be noted that a SIS is not required if the risk is already acceptable by non-SIS techniques such as operator response on alarms, safety distance, etc.

In offshore oil and gas industry, SIS prevents hazards on offshore installations and onshore processing facilities initiated by excursions of operating parameters such as pressure and level outside operating limit (Rothschild, 2004). This can be achieved by monitoring operating parameters and taking actions after detecting excursions.

Proper planning and management of SIS will obviously improve process safety (Summers, 2006). Furthermore the economic benefits can also be gained from appropriate development of SIS.

1.1.1 SIS versus BPCS

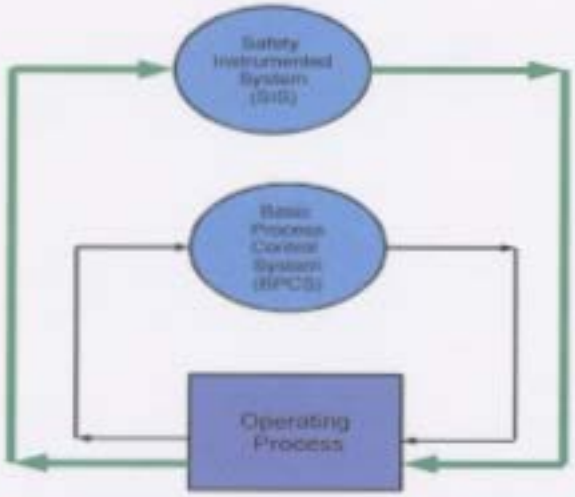


Figure 1 SIS VS BPCS

SIS is not BPCS. BPCS represents basic control system installed in most process systems such as level control system, flow control system, pressure control system, etc. Although BPCS and SIS are generally comprised of similar components, the objectives of the two systems completely vary.

BPCS can be considered as a positive system and is responsible to maintain or change process condition. It operates at all times and does not have diagnostic routines looking for faults. SIS can be considered as a passive system and acts like a safety guard of operating process. Almost all the process systems need BPCS. SIS is only required if process safety needs to be improved.

Because the aims of the two systems vary, separating SIS from BPCS can enhance safety ability of operating process system. In process industry, BPCS is almost always separated from the SIS, but the distinct functions of each can be part of an integrated system (Cusimano and DiNapoli, 2001).

1.1.2 A Simple SIS Example

Figure 2 shows a simple process system where the flammable liquid is drawn from a process source into a tank. A basic level control loop, which can be considered as a BPCS in this case, is provided in order to maintain liquid level in tank at 50% full. This control loop is composed by a level transmitter (LT1), a level controller (LC), and a final control element (valve FC). The tank also has a Pressure Safety Valve (**PSV**) and if the liquid level exceeds the threshold value (50% in this case) dangerous vapor will be emitted from PSV.

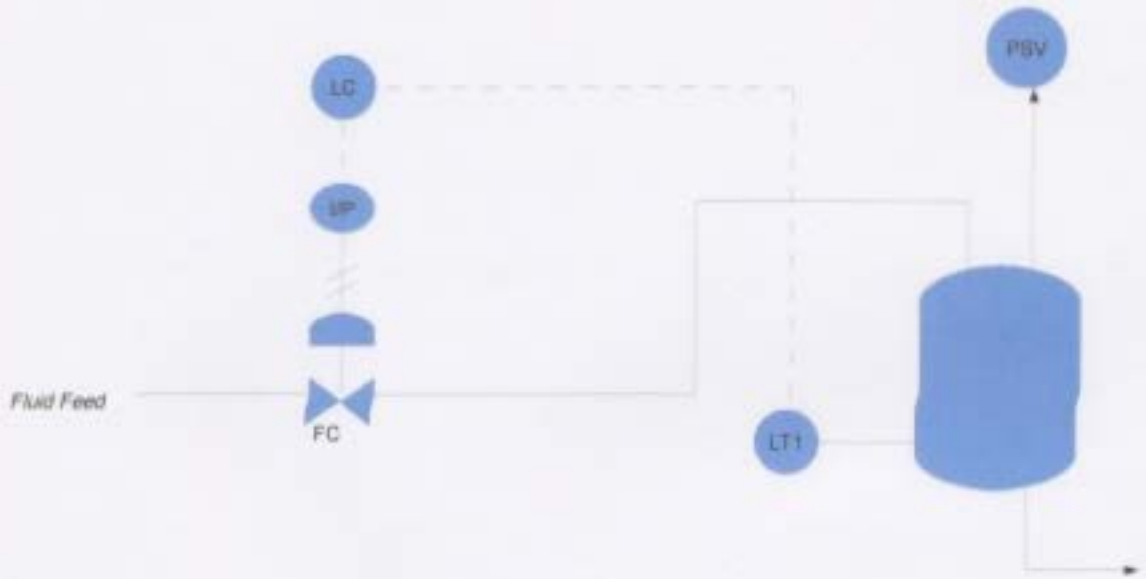


Figure 2 Basic tank level control without SIS

A simple Emergency Shutdown (**ESD**) system is added to this process (Figure 3). The purpose of this system is to shutoff the liquid feeding the tank. Another level transmitter (LT2) is set to detect liquid level in the tank. If extra high level of liquid is identified, logic solver will shutoff the valve (not same valve used by basic level control loop). This valve will remain closed until the defect in BPCS has been corrected. This system is not a part of control system. It is an extra system added for safety reasons and can be regarded as a simple SIS.

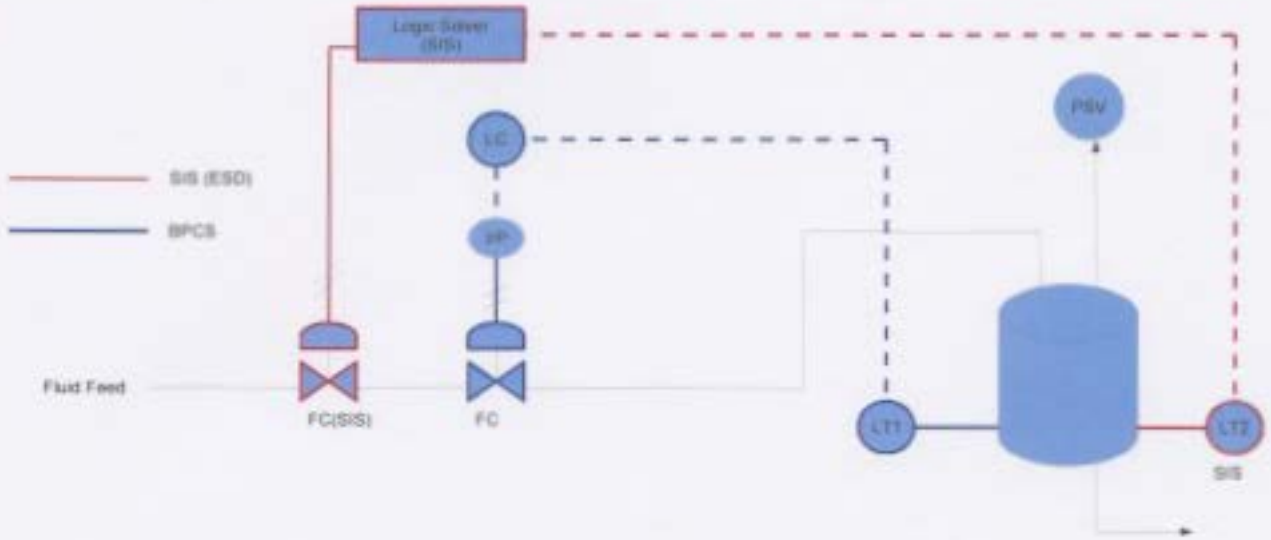


Figure 3 Basic tank level control with SIS

1.2 Safety Function (SF)

The IEC 61508 describes a safety function as "*function necessary to ensure the required functional safety for each determined hazard shall be specified*" (2000). This standard provides fundamental purpose of safety function: it is designed to implement the functional requirement of SIS.

IEC 61511 presents a more detailed definition of safety function: "*function to be implemented by a SIS, other technology safety-related system or external risk, reduction facilities, which is intended to achieve or maintain a safe state for the process, with respect to a specific hazardous event*" (2003).

In process industry, a safety function "is a set of specific actions to be taken under specific circumstances, which will move the chemical process from a potentially unsafe state to a safe state" (Marszal and Mitchell, 2003). This explanation of safety function has been utilized in various protection systems of process industry including ESD, high pressure separator, reactor protector, and other systems.

1.2.1 Purpose of SF

A safety function works as a protection against a specific and identified hazardous event. It is a method to define the functional relationship between inputs and outputs in SIS. Inputs can be regarded as sensors, outputs can be regarded as final control elements and safety function can

be regarded as a logic solver. In process industry, safety function is implemented by a collection of equipment pieces. Consider a heater fired on natural gas fuel. Excess fuel gas and loss of flame can lead to an explosion. An emergency switch is able to turn off heater to prevent this hazard. The procedure of this action can be regarded as a safety function.

Safety function only reduces risk (*probability* * *consequences*) and never completely eliminates the risk (Wiegerinck, 2002). However it would be sufficient to reduce the risk to an acceptable level. Reconsider the heater example above: the chance that the switch fails to close the heater does exist. Although the opportunity of the accident has definitely been reduced, it is not guaranteed that the explosion will not occur after adding an emergency switch. Actually in the beginning of designing safety function, the desired probability of failure needs to be considered.

A safety function can be subdivided into multiple sub functions according to all possible events that could lead to a specific hazardous incident. For example, a safety function is required to protect a reactor against overpressure. The cause of this hazard could be the loss of coolant, high pressure in reactor, or loss of service of ESD system. Each of those events can be prevented by a sub safety function respectively.

SF is able to assist SIS to reduce the risks. The amount of risk reduction is principally concerned. It can be measured based on the calculated Probability of Failures on Demand (**PFD**), which is the probability that SF fails to maintain safe state when predetermined safety conditions are violated. This probability can be estimated by calculating total probabilities of:

1) SF input services fail to tell the logic solver of SIS to take the action.
2) Logic solver was told to take action, however it fails to initiate it.
3) Final control-elements fail to take action when action is initiated by logic solver.

Since it is difficult to get the exact value of unsafe failure rates, safety analysis is used to measure the likelihood of SIS performing required safety functions.

1.3 Safety Analysis

There is no broadly agreed definition of safety analysis. The one proposed before is "safety analysis is a systematic procedure for analyzing systems to identify and evaluate hazards and safety characteristics" (MacDonald, 2003). This definition includes both quantitative and qualitative methods. The purpose of safety analysis is to improve systematic safety and reduce risk. It can be considered as a supplement to a company's own safety activities.

Safety analysis should be conducted before and after applying corresponding SIS to the systems. To obtain the confidence that the risks associated with safety requirement are acceptable, safety analysis is carried out to ensure that system design is consistent and maintain safe behavior.

Safety analysis is based on the knowledge of hazards from previous studies. The information and data is collected by a design engineer, who has a thorough knowledge of overall system and its design. Safety function can be considered as an approach to conduct safety analysis. Since a specific SIS can be represented by multiple safety functions according to all possible events, each SF can be used for the purpose of safety analysis.

Safety analysis is able to be used for judging the performance of safety function(s) of corresponding SIS. Safety Integrity Level (**SIL**) is developed as a quantitative evaluation factor of this judgment.

1.3.1 Safety Integrity Level (SIL)

Safety integrity can be understood as probability that a safety-related system will satisfactorily perform the required safety functions under all stated conditions within a given period of time (Kosmowski, 2006). SIL represents the amount of risk reduction that is required from a safety function. IEC 61508 defines SIL as "*a discrete level (one of four) for specifying the safety integrity requirements of safety function.*" (2000). Safety integrity level 4 (SIL4) is the highest level and safety integrity level 1 (SIL1) is the lowest one.

SIL has become increasingly part of the design and operation of safety instrumented system (Kirkwood and Tibbs, 2005).Companies are now specifying SIL based on the amount of risk

reduction that is required to achieve a tolerable risk level. The SIS is designed to meet or exceed this level of performance (Marszal and Mitchell, 2003).

SIL is calculated by the Probability of Failure on Demand (**PFD**) or Risk Reduction Factor (**RRF**). The IEC standard provides following table for SILs:

Table 1 Definitions of SILs for demand mode of operation from IEC 61511-1

SIL	Range of Averaged PFD	Range of RRF
4	$10^{-5} <= PFD < 10^{-4}$	100,000 >= RRF > 10,000
3	$10^{-4} <= PFD < 10^{-3}$	10,000 >= RRF > 1000
2	$10^{-3} <= PFD < 10^{-2}$	1000 >= RRF > 100
1	$10^{-2} <= PFD < 10^{-1}$	100 >= RRF > 10

SIL is intended to provide targets for developers. According to the value of SIL, developers can understand what the intended safety function is going to achieve and choose desired instruments to implement it. This needs to be done in early level of development stages in order to guarantee that proposed safety functions are realistic, achievable and affordable. The cost of SIS will increase if higher SIL is required.

Risk reduction terms can be applied to calculate RRF and PFD:

$$RRF = Fnp / Fp$$

$$PFD = 1 / RRF$$

where Fp = *Protected Risk Frequency*

Fnp=*Unprotected Risk Frequency*

The equations listed above are used in this paper to determine RRF and PFD.

There are several methods available to determine SIL of a safety function, which are described in IEC 61511.

- Fault and Event trees Analysis (FTA / ETA)
- Safety layer matrix method
- Risk graph
- Lay Of Protection Analysis (LOPA)

A constant method to determine SIL is required for any organization. Quantitative Risk Analysis (**QRA**), risk graph and LOPA are all established methods for determining SIL, particularly in process industry sector (Foord et al., 2004).

1.3.2 Layer of Protection Analysis (LOPA)

LOPA method was developed by American Institute of Chemical Engineering (**AICE**) as a method for assessing the SIL requirements of SFs (Foord et al., 2004).

LOPA is a semi-quantitative risk analysis method because this technique does use numbers and generate a numerical risk estimate. However, the numbers are selected to conservatively estimate failure probability, usually to an order of magnitude level of accuracy, rather that to closely represent the actual performance of specific equipment and devices (Hendershot and Dowell, 2002).

*LOPA implements the multiple Independent Protection Layers (**IPL**) to safeguard a process*, which is often used in process industry. Figure 4 illustrates the concept of protection layers.

Figure 4 Protection layers (onion model)

IPL represents the layer of protection against an initiating cause leading to an impact event. Specifically it might be:

- **General Process Design**: There may, for example, be the design of system that reduces the probability of a hazardous event.
- **Basic Process Control System (BPCS)**: Failure of a control loop is likely to be replaced by an extra control loop. Automatically control system is always added into the process to prevent undesired events.
- **Alarms**: Independent of the BPCS, an alarm might exist providing sufficient time for an operator to respond and take an effective action.
- **Safety Instrumented System (SIS)**: If all non-SIS IPLs above could not reduce the frequency of consequence (hazard) to an acceptable level, a specific SIS is implemented to meet the target object.

Figure 5 shows the concept of each IPL acting as a barrier to reduce the frequency of the consequence.

Figure 5 IPLs in LOPA

LOPA provides a consistent basis for judging if there are sufficient IPLs to control the risks of an accident for a scenario (Wiegerinck, 2002). If it is not enough, then more IPLs are added.

Table 2 shows the general form of a LOPA table:

Table 2 General format of LOPA table

Consequence & Severity	Initiating Event	Initiating Event Challenge Freq /yr	Independent Protection Layers Probability of Failure on Demand (PFD)				Mitigation Independent Protection Layers (PFD)	Mitigated Consequence Freq /yr
			Process Design	BPCS	Alarms	SIS (SIF)		

LOPA estimates the probability of undesired consequence by multiplying frequency of the initiating event by the product of the PFDs of each applicable IPL using the equation below:

$$f_i^C = f_i^I \times \prod_{j=1}^{J} PFD_{ij}$$

$$= f_i^I \times \mathrm{PFD}^{i1} \times \mathrm{PFD}^{i2} \times \ldots \times \mathrm{PFD}^{iJ}$$

where f_i^C = frequency for consequence C for initiating event I (Mitigated)

f_i^I = initial frequency for initiating event I

PFD^{ij} = probability of failure on demand of jth IPL that protects against consequence C for initiating event i.

The IEC61511 part 3 Annex F includes suggested or typical PFD values for factors such as operator responses and alarm system integrities (MacDonald, 2003).

LOPA is a relatively quick and straightforward approach for modeling the protected system. However LOPA may be inadequate if required component failure data is not available or failures are not independent. FTA can be used in these situations since this method can evaluate compound failures.

1.3.3 Fault Tree Analysis (FTA)

Fault trees originated in the aerospace industry and have been used extensively by nuclear power industry to quantify and quality the hazards and risks associated with nuclear power plants (Crowl and Louvar, 1991). It is the most common of quantitative analysis techniques to be used for detailed SIL determination (Kirkwood and Tibbs, 2005).

Fault Tree Analysis (**FTA**) is a graphical technique that provides a systematic description of the combinations of possible occurrences in a system, which can result in an undesirable outcome (Kirkwood and Tibbs, 2005). This analysis starts from a top event, an accident, and works backwards to various scenarios that can cause the accident (Rothschild, 2004). FTA is designed to accurately evaluate compound failures and account for any dependencies between failures.

FTA visually produces an additional failure map, which assists the developers and safety analyst to identify the strength and weakness of the whole system. Although FTA is a more robust technique than LOPA, it also has its own limitations:

1) FTA is a highly specialized study technique, which requires developers to have a thorough understanding of the target system. For some complex process systems, expensive software is required to conduct the FTA.
2) Comparing with LOPA, FTA takes more time to complete. The number of intermediate events related to one basic event could be large. This will make it more difficult to develop fault tree.

1.3.4 Combining FTA with LOPA

Because both LOPA and FTA have their own weaknesses and strengths, it is more convenient to combine these two methods in order to determine the SIL. This enhanced method integrates the simplicity of LOPA with analytical strength of FTA and provides a relatively quick and more efficient safety analysis. Combining LOPA with FTA needs the developers to consider the target process system as the safety related system. The concept of the overall safety lifecycle is crucial to system design. Each safety layer must be defined and developed independently.

The existing process system is first divided into various protection layers using LOPA onion model. SIS can be considered as an IPL. Other layers are determined according to the developer's knowledge and experiences for whole system. The procedure to perform the safety analysis using SIL, illustrated in Figure 6, can be summarized as followed:

1) **Determining target SIL:** It depends on the type of system such as continuous system or on-demand system. SIL2 is sufficient for most industry process systems.
2) **Adding IPL:** An alarm and a SIS can be added as an IPL.
3) **Calculating SIL by FTA:** Build fault trees based on the information from previous hazard study such as Hazard and Operability (**HAZOP**).
4) **Calculating RRF:** RRF can be calculated via dividing Fnp by Fp. If obtained RRF does not arrive at defined range according to the Table 1, then add more IPL and repeat all the steps described above.

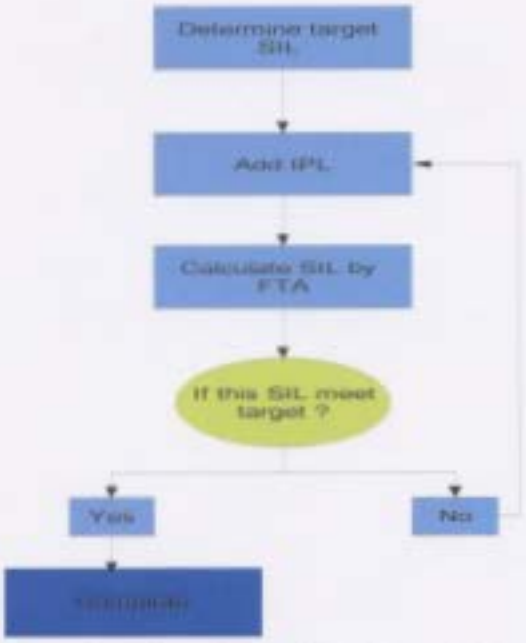


Figure 6 Procedure of determining SIL by combining FTA with LOPA

The detailed demonstration of this procedure will be discussed in a case study below.

1.3.5 Case Study of Safety Analysis Using Combining Method

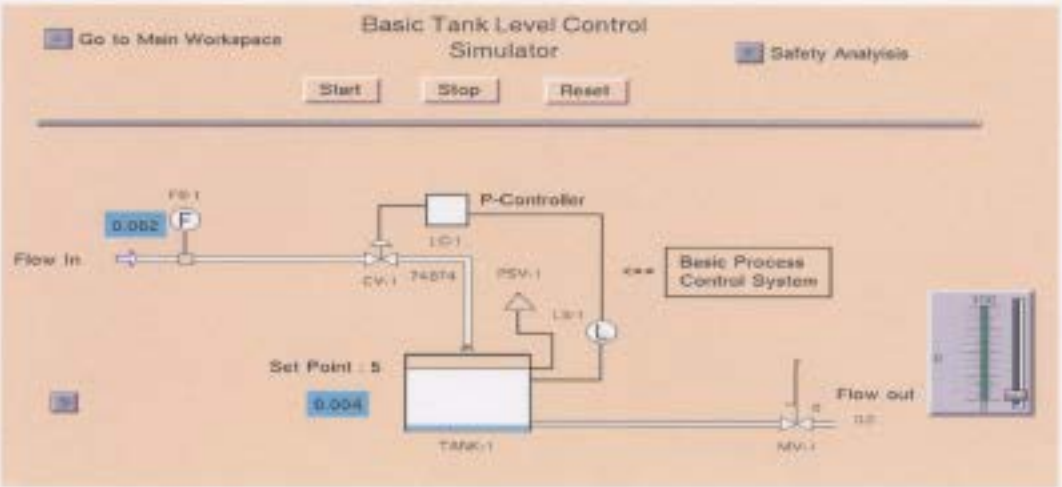


Figure 7 Snapshot of a storage tank simulator

Figure 7 shows a snapshot of a basic tank level control system simulator developed on G2 platform. The flammable liquid is drawn from a process source into a tank. The height of tank is 10 meters. A typical level control loop, which uses proportional controller, is provided in BPCS to maintain tank level at 5 meters. An explosion will happen if the level control fails for any reason and tank becomes full. A pressure relief valve (PSV-1) is installed. If the liquid has to escape from the tank through this valve, a dangerous vapor cloud will be formed. Failure rate data used in this case study are average values determined at a typical chemical process facility and selected from the book by Frank P.Lees (Frank, 1986).

The reasons for loss of level control include:

- Failure of Level sensor LS-1
- Failure of proportional controller SC-1
- Control valve CV-1 fails to close

The failure of unprotected system (with only BPCS) is 0.01635 per year and the target SIL is level 2. According to the Table 1, the acceptable range of RRF should be between100 and 1000.

1) Unprotected system

The failure of unprotect system is 0.01635/year.

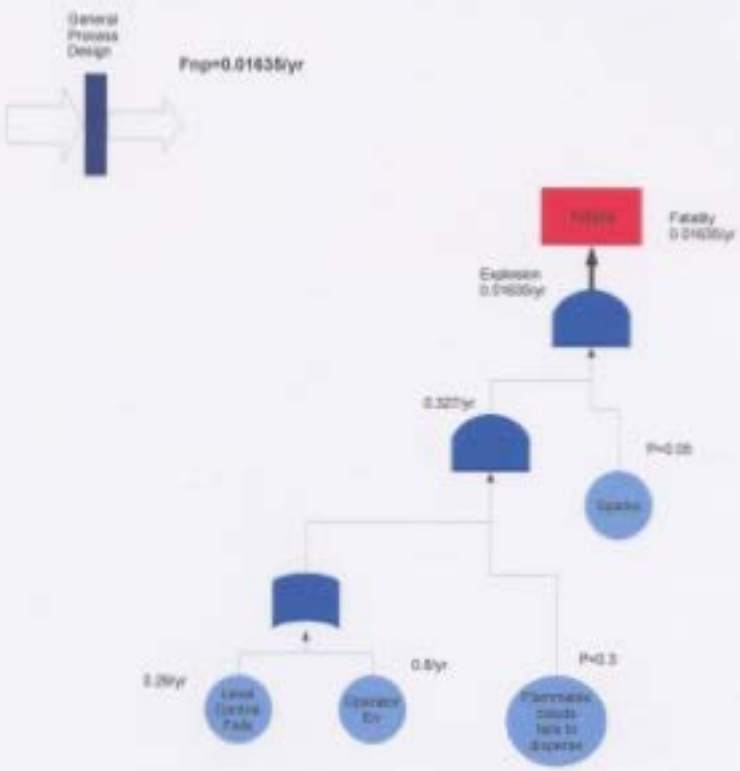


Figure 8 FTA tree and LOPA analysis representing unprotected system

2) Add non-SIS IPL (operator response to a alarm)

A high level alarm is then added to the system. This alarm will be activated if liquid level in tank reaches 6 meters. After adding a non-SIS layer, Fp (protected risk frequency) is 0.001635/yr.

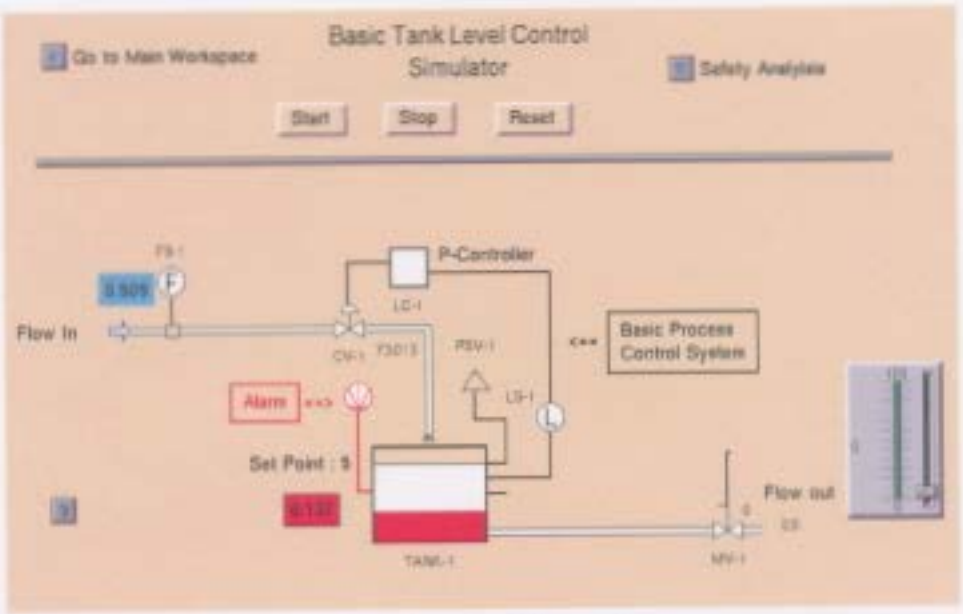


Figure 9 Snapshot of a storage tank simulator after adding an alarm

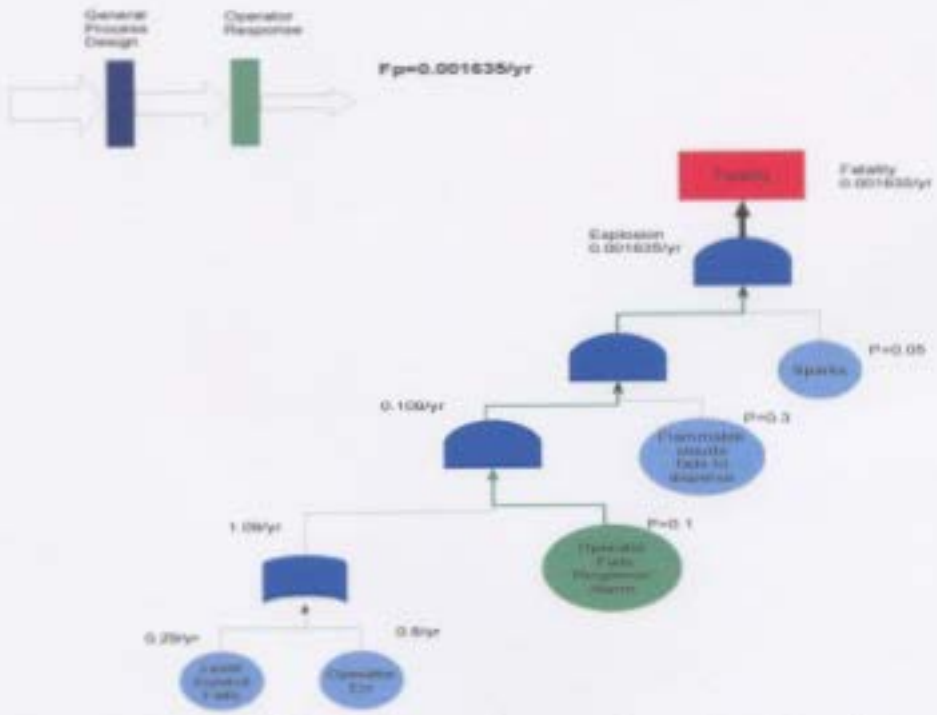


Figure 10 FTA tree and LOPA analysis representing protected system after adding an alarm

$RRF = Fnp / Fp = 10$

$PFD = 1/RRF = 0.1$

Both RRF and PFD have not arrived at defined range according to Table 1. More IPL is needed to meet the design object.

3) Add SIS IPL

A SIS, which is a simple ESD system in this case study, is added to the system. This SIS is composed of a level sensor, controller and a solenoid valve. When the alarm is activated and no operator open valve MV-1 after some period, the SIS will automatically close valve SV-1. Therefore, whole system will be shut down until operator recognizes this hazard. F_p now becomes 0.0001226/yr.

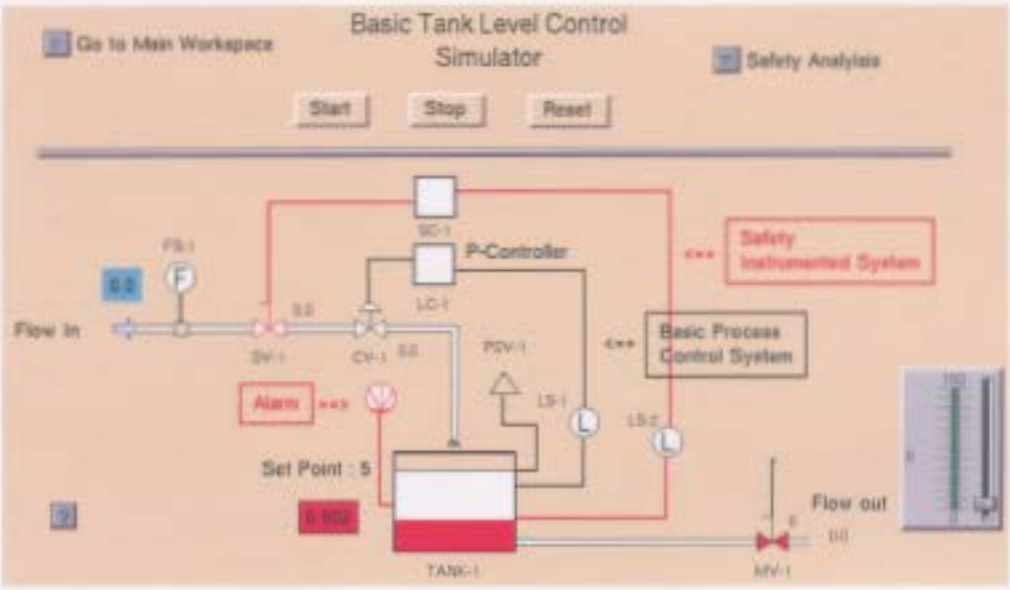


Figure 11 snapshot of a storage tank simulator after adding a SIS

$RRF = Fup / Fp = 133$

$PFD = 1/RRF = 0.0075$

Since RRF is between 1000 and 100, final SIL meet the target SIL. SIS is accepted and no more IPL is needed.

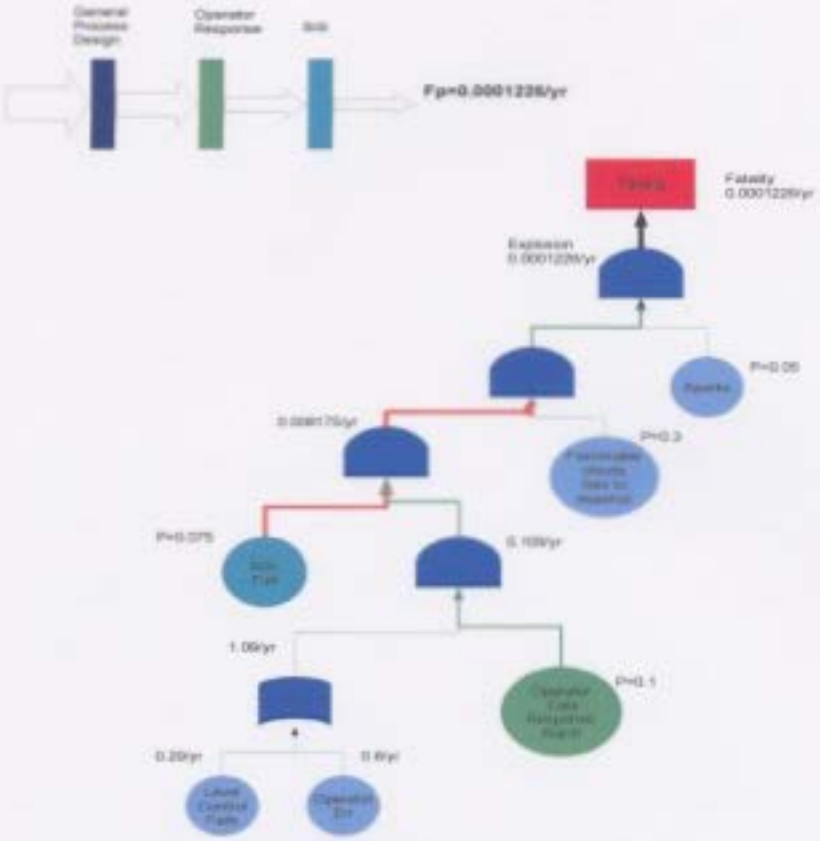


Figure 12 FTA tree and LOPA analysis representing protected system after adding a SIS

From case study above, it can be seen that the combining method is straightforward, intuitive, and even easy to conduct. It also provides a visual failure map, assisting the analyst in better understanding the system and building safety systems.

1.4 Objectives of Present Work

The preceding discussion introduces the concepts and significances of safety instrumented system, safety function, and safety analysis. A new safety analysis method, combining LOPA with FTA, is also presented. This new method can be carried out to determine the performance of implemented safety instrumented system.

Motivated by the significance and requests of high performance SIS for process industry, a methodology for fault diagnosis based on real-time data is proposed in this research. In addition, a computer-aided tool based on this methodology is also developed.

The objectives of this research include:

- To propose a methodology for real-time fault diagnosis in process system and its use in developing real-time SIS
- To implement the proposed methodology by developing a computer based tool
- To study and evaluate the performance of the proposed methodology using developed tool

1.5 Organization of the Thesis

This thesis is divided into seven chapters. The first chapter introduces the concept and significance of safety system, safety function, and safety analysis. It also introduces the current safety related practices in process industry. In addition, an improved safety analysis method, which combines LOPA with FTA, is presented with a detailed case study.

Chapter 2 presents review of the available approaches and a proposed safety system development methodology, which is divided into three stages including system simulation, knowledge-based fault diagnosis method and G2 application development. In chapter 3, a micro steam power unit is modeled using the simplified process model and simulated by G2 expert system. Chapter 4 proposes a knowledge-based fault diagnosis method which is comprised of three steps: acquiring information, making inferences and taking actions. A computer application is also implemented based on three steps of this method using G2's GDA application. In chapter 5, application of the proposed methodology and tool is demonstrated using simulated and industrial data. Chapter 6 discusses the results obtained from chapter 5. Chapter 7 includes the final conclusion and future possible works of this research.

Chapter 2

METHODOLOGY FOR DEVELOPING REAL TIME SAFETY INSTRUMENTED SYSTEM

2.1 Review of Available Approaches

The standards and guidelines for designing SIS have been studied and developed for almost 20 years. In 1987, UK HSE (UK Health and Safety Executive) published an excellent document on the use of Programmable Electronic Systems for use in safety applications. In 1993, AICE's (American Institute of Chemical Engineers) center for Chemical Process Safety released the book, "Guideline for Safe Automation of Chemical Process", which covers the design of DCS (Distributed Control Systems) and shutdown systems. IEC (International Electrotechnical Commission) has also been working on the overall standards for all industries for more than 10 years and published IEC 61508 and 61511. These two documents cover the use of relay , solid state and programmable systems and will apply to all industries such as medical, transportation , nuclear, etc. It should be noted that all of the standards and guidelines listed above are performance oriented, not prescriptive (Gruhn, 1999).

All of the standards and guidelines listed above provide the fundamental requirement and instruction of designing SIS. According to these standards, Gryhn (1999) introduced a basic SIS design life cycle, which is illustrated in Figure 13. These procedures have been accepted for most of industry sectors as the procedures to follow during SIS concept design stage. ABB's System 800xA, a high-integrity modular controller, is a complete IEC 61508 and IEC 61511 compliant SIS and has improved process availability (McMath and Kingman, 2005).

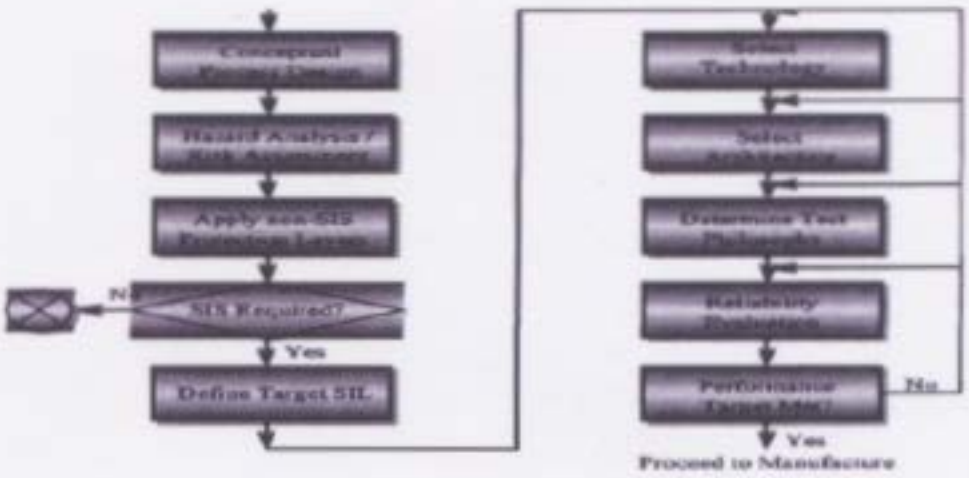


Figure 13 SIS design life cycle (Gruhn, 1999)

As mentioned in the previous chapter, developing a safety instrumented system can be regarded as designing one or more corresponding safety functions. IEC 61511 has already introduced the functional safety standard, which concerns the safety concept for the process industry (Kosmowski, 2005). Frankauser (2002) compared safety functions with control functions and clarified the conflicts. Marszal and Mitchell (2003) provided a discussion for defining safety functions for various process systems and hazards. These safety functions are very specific and only designed for the designed SIS.

Furthermore, computer technology has been applied to safety system design since 1990s. In 1993, Goring C. J. developed a Triple Modular Redundant (TMR) computer-based safety control systems for the North Sea offshore production industry (Goring, 1993), which includes fire/gas system and ESD systems. These systems can only be considered as the simple SISs and major safety functions are all after-event functions. Their primary purpose is to shut down the process plant if process operating out of bounds is detected. This type of safety instrumented system is not acceptable for current complex and unpredictable process systems, a more intelligent and accurate system is needed.

2.2 Proposed Methodology

There is one general safety function which can be considered as a principal part of each SIS. This function is called *fault diagnosis function.* The function of diagnosis is among the objectives of the monitoring and falls under a total process of supervision (Sharif and Grosvenor, 1998). The purpose of this function is to monitor the process through the real-time information from the lower level (sensors) and take actions on higher level (controllers).

To develop a real-time SIS, fault diagnosis function might be a common approach which can be applied to detect over-all faults and even faults in components. The goal of this chapter is to propose a general methodology to develop the SIS by designing general fault diagnosis function, which can be used in various process systems.

The proposed methodology implementation can be divided into three stages, which is illustrated in Figure 14:

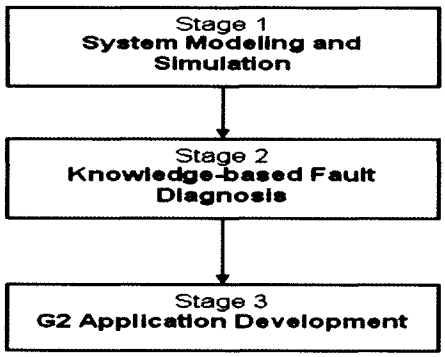


Figure 14 Three stages of proposed methodology implementation

2.2.1 System Modeling and Simulation

Simulation has been used successfully for years to improve system design and management (Harrell, 1998). It is a quantitative technique that examines the detailed execution of the process at a higher level. Many developers have been benefiting from the use of simulation to analyze and improve their designs.

During the step of designing SIS, it is more flexible and applicable for a developer to have a complete system simulator based on developed system model as a platform rather than trying

to apply any extra system into real process system. Most process systems are dynamic and unpredictable. Directly modifying current system components could cause potential threat to life or environment.

Developing system simulation can be divided into four stages, which are outlined below:

- **Specifications:** This is the stage of conceptual design. The overall system as well as various components is required to be analyzed. In addition, the interfaces between systems and subsystems need to be defined and validated.
- **Modeling Design:** At this stage, detailed system/component models are designed according to the system behaviors. The available modeling methods include state space, classical empirical model, and other mathematical methods. This stage could be challenged since it is difficult to model some non-linear systems precisely. The acceptable modeling accuracy should be considered before carrying out this stage.
- **Development:** The simulator, which is based on the designed model, is developed in this stage. Since the desired simulator is computer-based, the appropriate software platform should be chosen. The possible options include *HYSYS*®, ASPEN *Plus*®, Java, G2, etc.
- **Verification:** The developed simulator needs to be verified in order to meet the requirements. This can be performed by comparing the outcome of simulator with real system output.

The details regarding system simulation are described in Chapter 3.

2.2.2 Knowledge-based Real-Time Fault Diagnosis Method

Faults, also referred in industry as critical conditions or abnormal situations, are a range of abnormal operating states that are beyond a normal state, but fall short of automated shutdowns (Siegel et al., 2004), such as those that take place during an emergency. Typically these conditions are the consequences of combinations of events that are unexpectedly occur at same time. Fault is also understood as any kind of malfunction in the actual dynamic system (Mohamed & Ibrahim, 2002). Such malfunction could result from process variables, process

components, or even basic control systems. If the system behavior is regarded as malfunctioning, then appropriate fault diagnosis mechanism can be used to detect this faulty activity.

The method of fault diagnosis began in various places in the early 1970s and has been receiving more and more attention over the last two decades (Dash et al., 2004). The increasing interests are applied for two major applications: academic and industrial application due to safety related matters. The detection and diagnosis of faults in process systems is of great significance. An early detection of faults may help to avoid incidents, process upsets, product deterioration, performance degradation, and damage to human health or even loss of lives (Wolfram et al., 2001). In this paper, fault diagnosis method is implemented to realize the fault diagnosis function of developed safety system.

Fault diagnosis can be performed by employing different approaches such as model-based and knowledge-based. Model-based approach uses quantitative models and equations to estimate the states or parameters of the system. However, in practice it is almost impossible to obtain a model that exactly matches the process behavior (Mohamed & Ibrahim, 2002). The mismatch between the behavior of the model and the plant may lead to large error signals (Howell, 1994), which can cause false alarms unless appropriate thresholds are used. Furthermore, it can be impossible to describe some non-linear systems by analytical equations. These disadvantages increase the necessity of using an alternative approach: knowledge-based approach.

Knowledge-based fault diagnosis is performed based on the evaluation of on-line monitored data according to a set of rules which the human expert has learned from past experience (Monsef et al., 1997). The knowledge includes the locations of input and output process variables, patterns of abnormal process conditions, fault symptom, operational constraints, and performance criteria. The operator and engineer's intelligence related to the specific process systems are implemented into this approach. Their knowledge can help to recognize the potential faults based on previous experiences. This approach can reduce the burdens on exact numeric information and automates the human intelligence for process supervision (Lo et al.,

2006). Compared with model-based approach, it is particularly suitable for large industrial plants since those non-linear real plants are extremely difficult to model and linear approximation of the model will introduce large errors in the results. In addition, knowledge-based approach is able to reduce the complexity of implementing the corresponding safety system and make it flexible and easy to understand and follow. Combining knowledge-based fault diagnosis method with real-time process variables monitoring will improve the efficiency and reliability of detecting fault behavior and overall effectiveness of the system.

Motivated by the advantages mentioned above, a knowledge-based real-time fault diagnosis method is proposed in this thesis for the purpose of implementing general fault diagnosis function in SIS, which is comprised of three steps (Figure 15). The first step is the acquiring of the real time process information, from critical equipment pieces, such as boilers, compressors, separators or reactors. Temperature, pressure, level, and flow rate are the most important process variables to be monitored and have the capability of representing the state of operation in a variety of equipment pieces. The fault in these variables can affect the stability and safety of the whole process system. The second step is making inferences (diagnosis) based on acquired process information. The last step is making actions according to the inference values, such as informing operators, raising alarms, shutting down equipment, activating higher layer protections and trying to bring the system back to normal condition. The details of this method are described in Chapter 4.

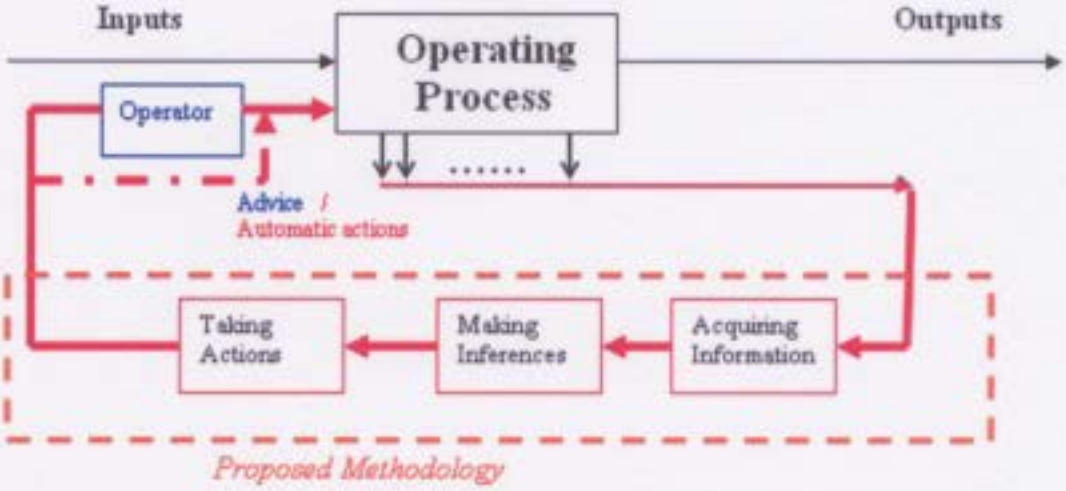


Figure 15 Three steps of proposed fault diagnosis method

2.2.3 G2 Application Development

Fault diagnosis, both in terms of its likelihood and the likely period of occurrence, remains a particularly difficult task. In the environment of real-time data collection, high volume of information needs to be analyzed and processed in a short period. Final outcome based on real-time data analysis is likely to be reasonable and relatively precise only through the assistance of some computer aided tools. Furthermore, modern process plants have become more complex and involved using high technology machines and computers. To develop an acceptable diagnostic system for such plants, it is important to consider the underlying principles of fault diagnosis in general, and try to develop a computer based fault diagnosis system which satisfies the specifications and demands (Sharif and Grosvenor, 1998).

The aims of developing computer application are to provide the platform for testing proposed methodology, enhance the performance of the safety system, and decrease the response interval between detecting fault and taking further actions. In order to ensure that all safety goals have been achieved, developed computer application should have the following capabilities:

- **Rule-based reasoning:** Turning complex data into useful information by reasoning about it through object models and rules.

- **Event detection:** Continuously monitoring multiple asynchronous events especially for abnormal events and data streams simultaneously in real time.
- **Diagnosis capabilities:** Ability to realize complex diagnostic procedures.
- **Connectivity:** Bridges to numerous standard databases, process control systems and user development environment.
- **Integrated intelligent technologies:** Ability to implement fuzzy logic, neural networks.

G2 software from Gensym Corporation has the abilities to perform these capabilities. Founded in 1986, Gensym Corporation is a leading provider of rule engine software and services for mission-critical solutions that automate decisions in real time. Gensym's flagship G2 software applies real-time rule technology for decisions that optimize operations and detect, diagnose, and resolve costly problems.

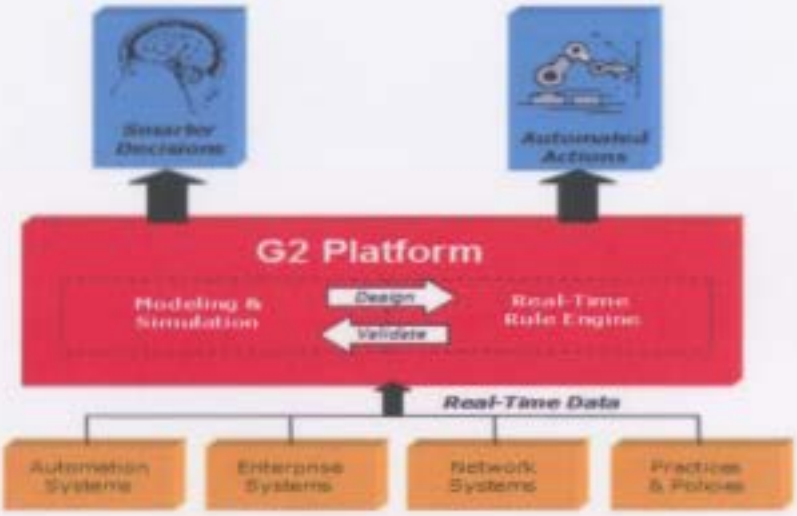


Figure 16 G2 platform (www.gensym.com)

G2 is one of the world's leading real-time engine platform and uniquely combined real-time reasoning technologies including rules, object modeling simulation, and procedures in a single development and deployment environment. Real-time data is transformed into automated decisions and actions in real time by G2 platform. A wide range of industry solutions are

supported by G2 including oil and gas industry, process manufacturing, power utilities, aerospace, and telecommunication.

G2 supports three basic types of programming constructs for processing data: rules, methods and procedures. One of the primary implemented processing methods in this simulating platform is rule-based processing since it has two general techniques. Another method is procedural processing, which includes single-threaded processing and multi-threaded processing.

GDA, the *G2 Diagnostic Assistant*, is an environment for developing and running intelligent operator applications. Its principal component is a graphical language that lets user express complex diagnostic procedures as a diagram of blocks, also called an Information Flow Diagram (**IFD**). These blocks are connected by paths that show how data flows through the diagram.

A GDA application contains various schematic diagrams, which have capability of

- Acquiring data from real-time processes
- Making inferences based on the data
- Taking actions based on the inference values, such as raising alarms, sending messages to operators, or concluding new set points

Motivated by the G2 platform introduced above, the proposed methodology including system simulation and fault diagnosis method are implemented on the platform of G2 expert system using GDA applications. The details of this method are described in section 3.3 and section 4.5.

Chapter 3

SYSTEM MODELLING AND SIMULATION

System modeling and simulation is the first stage of the proposed methodology. The purpose of this stage is to generate the input data and provide a platform for designing and testing fault diagnosis function. It is based on the created system model with the capability of describing the system behaviors correctly under normal or abnormal conditions.

In this chapter, a real process system is modeled, simulated by G2 software and discussed. The stability and feasibility of model is also verified through comparing data generated by simulation with history records of the unit.

3.1 Modeling Design

In this research, a micro steam power unit, shown in Figure 17, is first chosen to be studied. It is located in the thermal lab in the engineering department of Memorial University and is used for the purpose of student experiments. The schematic of this power unit is shown in Figure 18.

Figure 17 The picture of steam power unit

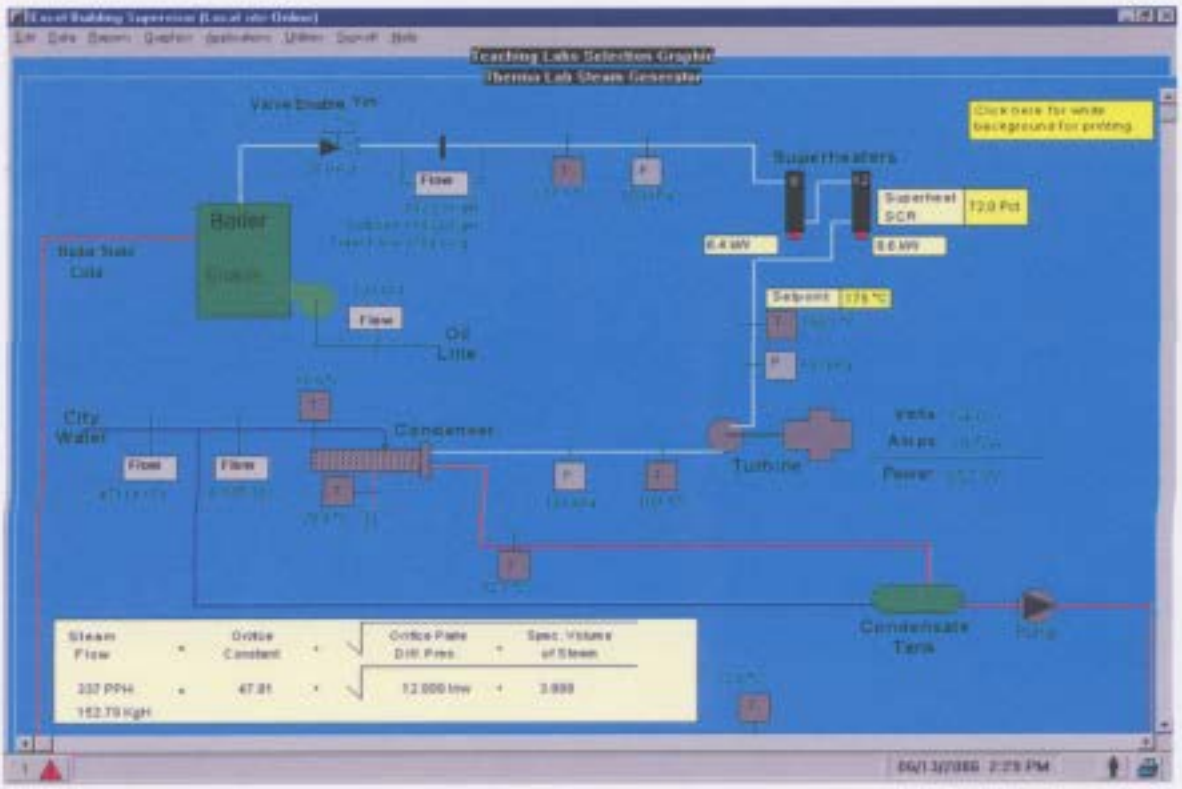


Figure 18 The schematic of the steam power unit

This power unit is comprised of several units including a steam boiler, super heaters, steam turbine, condenser, condensate tank, pump, and other control system components. Steam is first generated in the boiler and moves along the system to run the turbine. Finally it powers ten electric bulbs. The unused steam is condensed by condenser and deposited in the condensate tank for the future use.

Several system modeling methods can be used such as state space and classical empirical model. However the complexity of whole thermal system makes it difficult to identify and model by a conventional approach. Furthermore it is not necessary to build a highly precise system model for this project. The accuracy can be considered as acceptable level, which indicates that certain of errors are allowed. Due to reasons mentioned above, it will be more flexible and efficient if a straightforward modeling approach is implemented. Therefore a simplified process model is developed by studying the behaviors of process variables and thus identifying the relevant response equations.

In this power system, three major process variables are pressure, temperature, and flow rate. Steam pressure in the boiler is the key variable of the system. It is the starting point to determine other variables such as temperature in the boiler, flow rate of steam running out of the boiler, power generated by the steam turbine, and so on. Two steps have been defined in order to determine corresponding model response equations: identifying the order of the process model according to process components behavior and determining model parameters based on time response of the unit system model.

3.1.1 System Model Order Identification

Figure 19 shows the trend chart of boiler steam pressure recorded every minute. The steam pressure starts to increase from 200 kPa and reaches steady state when it becomes close to 700 kPa. Since it is similar to the response of a typical second order system, it can be modeled using equation below:

$$f(t) = 1 - \frac{\exp(-\xi\omega_n t)}{\sqrt{1-\xi^2}} \sin(\omega_n \sqrt{1-\xi^2}\, t - \tan^{-1}(\frac{\sqrt{1-\xi^2}}{\xi}))$$ (Ogata, 2004)

where: ω_n = undamped natural frequency

ξ = damping coefficient

Other system components also need to be modeled, such as pipe time delay, valve dynamic and superheater dynamic. Since these components only delay the response time of the particular unit, they can all be modeled as first order systems whose equation is

$f(t) = K(1 - e^{-t/\tau})$ (Ogata, 2004)

where: τ = time constants

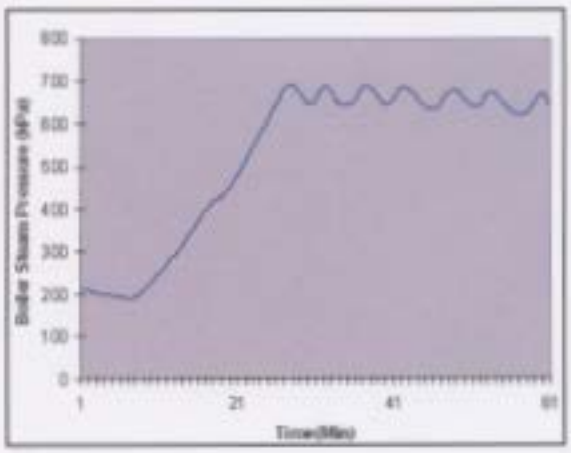


Figure 19 Trend chart of boiler steam pressure

3.1.1 System Model Parameters Determination

The order of the process model provides the structure of the equation. In order to complete the simulations the equation parameters must be determined. System characteristics, such as rising time and overshoot, can be obtained by analyzing trend log records. After that, the desired equation can be obtained, tested, and adjusted using the Matlab. For example, System characteristics can be determined by studying trend chart in Figure 19. Overshoot (OS) is 0.073 and rising time (Tr) is about 20 minutes (1200 seconds). Therefore undamped natural frequency ω_n and damping coefficient ξ in second order system equation can be obtained using observed overshoot and rising time. The calculated equation is shown below:

$$f(t) = 1 - \frac{\exp(-0.002176 \times t)}{0.7684} \sin(0.0026 \text{b} \times t - 50.2)$$

Figure 20 shows a simple simulation of steam pressure using a second order equation in Matlab using obtained equation. Two diagrams (Figure 19 and 20) have very similar patterns with little variation. This is because the two diagrams use different time axis and there is no noise simulation in Matlab. However, not all models can be adjusted by this method. Some process components, for example, pipe delay and unit dynamics, have no chart to compare. In order to deal with this situation, parameters must be decided according to the developer's understanding of the entire system.

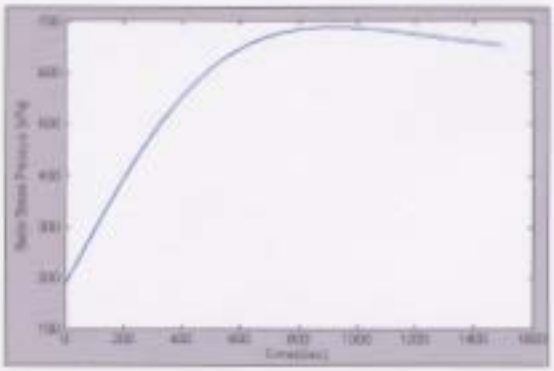


Figure 20 The simulated boiler steam pressure by matlab

3.2 System Simulation using G2

The purpose of the individual model simulation is to check and validate various models (Mouss et al., 2004). After deriving all determined model equations (Appendix C) from the previous stage, establishing the complete desired simulator becomes feasible.

Figure 21 illustrates a snapshot of the simulator developed in the G2 Shell (process simulator).

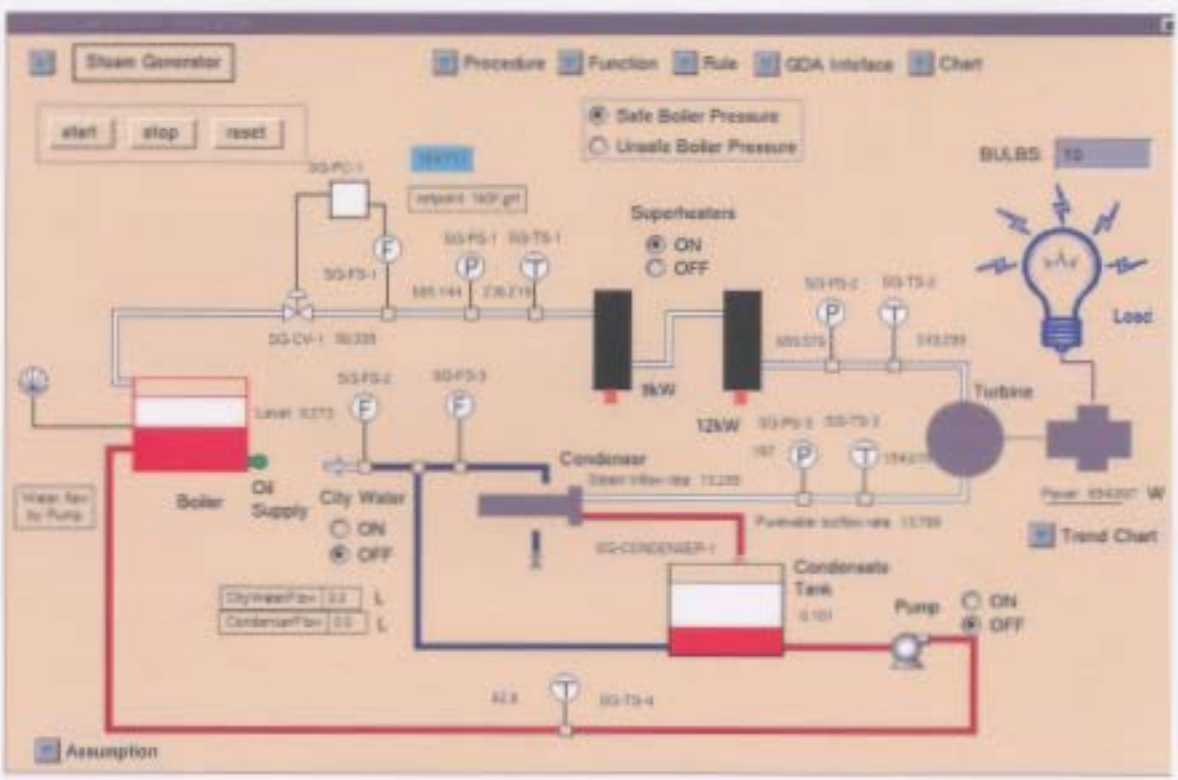


Figure 21 The snapshot of simulator in G2 shell

The development of this process simulator is divided into 4 steps:

1. G2 object definition

Each component and unit of the power system must be defined and represented as a G2 object. The attributes of the object are specified according to the knowledge and experience of the developers. For example, the attributes of a steam boiler object should at least include steam-outflow-rate, water-inflow-rate, steam-temperature, and boiler-pressure. These attributes are all essentially necessary to describe how this object functions.

2. Model equation definition

The G2 function is used to represent the obtained model equation. This makes it easy for developer to modify the characteristic of the desired model without changing procedure code. All functions are defined in the time domain. In order to increase or decrease the speed of operation, the simulation time period need to be adjusted. Figure 22 illustrates an example of G2 function, which represents the boiler pressure model equation.

```
simulate-blr-pres-calculate(t)=(1 - sin(wn *
    sqrt(1 - expt(omega,2)) * t + arctan
    (sqrt(1 - expt(omega,2))/ omega)) *
    exp(-1 * omega * wn * t)/sqrt( 1 -
    expt(omega,2)))* BLR-PRESSURE-SP +
    pres-white-noise
```

Figure 22 An example of G2 function

All the developed G2 functions have been included in Appendix C.

3. G2 procedure development

To generate an acceptable process simulation environment, simultaneous operations must be implemented. The G2 procedure program can execute a sequence of independent actions. Therefore, they are capable of simulating real-time process operations. Basically a procedure is

responsible for dealing with the input and the output of each object. Each system unit, which has already been defined by a G2 object, has its own procedure associated with its own respective attributes. Boiler procedure, which aims to control and synchronize other procedures, is the central foundation of other procedures. The actions included in this procedure can be summarized as follows:

1) Check the initial process conditions, for example whether or not the super heater is powered up
2) Calculate boiler pressure based on current simulation time
3) Calculate boiler steam temperature and flow rate based on obtained pressure
4) Calculate water level inside boiler
5) Calculate temperature and pressure before and after passing through the steam turbine
6) Update all sensor readings
7) Calculate turbine power
8) Start other procedures to calculate more process parameters such as the flow rate of the condenser and the water level in the condenser tank
9) Wait 5 seconds and repeat all the calculations described above

The details of G2 procedure program are described in Appendix A.

4. G2 rule definition

The G2 rules have been developed to detect specific process events. Unlike the G2 procedure, well defined rules are designed to handle the process system actions after expected or unexpected events.

```
whenever the boiler-pressure P of sg-blr-1
    receives a value then conclude that the
    p-output of sg-ps-1 = P +
    random(-.025,.025)*P and  conclude that
    sg-ps-1 =the p-output of sg-ps-1
```

Figure 23 An example of G2 rule

In this simulator, the rules are also responsible for updating the simulated sensor outputs. If a procedure updates a process variable, the sensor associated with this variable also updates its output. One example of G2 rules is illustrated in Figure 23, which is designed to update simulated pressure sensor sg-ps-1.

All the developed G2 rules have been included in Appendix D

3.3 Simulation Verification

In order to verify the performance of the model and simulation, the outcome of simulation should be compared with unit history log records. To be efficient, the comparison must be based on similar process conditions, for example, normal or abnormal conditions. Three major process variables, which include boiler steam pressure, boiler steam flow rate, and turbine power, were selected for the comparison.

Figures 24 and 25 demonstrate both simulated process outcomes and unit history record charts. The simulation was conducted under normal process conditions.

Other non-daily operations, such as changing generator load and shutting off super heaters, can also be applied to the simulator in order to prove the accuracy of model.

Figure 24 The simulated process outcome

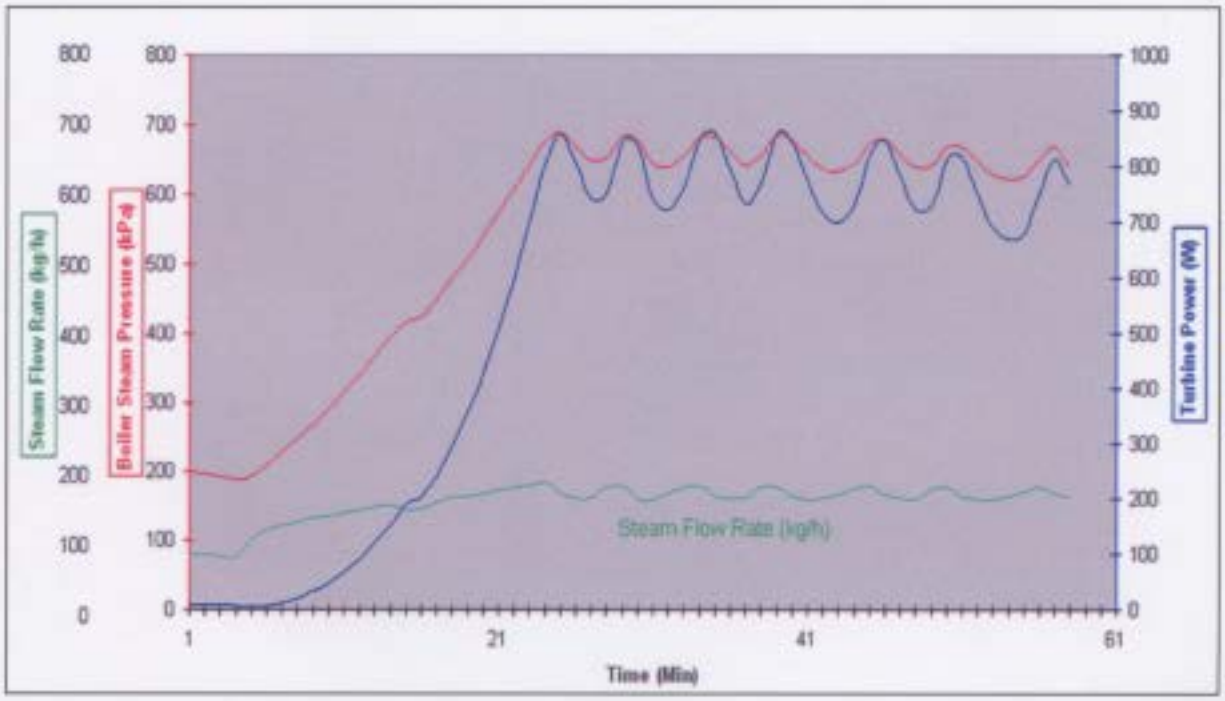


Figure 25 The unit trend record during normal operation

Figure 26 illustrates the result of the simulated process after reducing power load by using 8 bulbs instead of 10. Due to the noise and the limitations of the equipment, turbine power does not decrease to the expected level immediately. It starts to increase at first and drops to the

relevant level after a period of time. This event can be observed from the marked circle of Figure 27, a chart of unit trend records during this non-daily operation. The developed model fails to explain this situation. An additional model, which is designed to simulate this uncertain power noise, has been included.

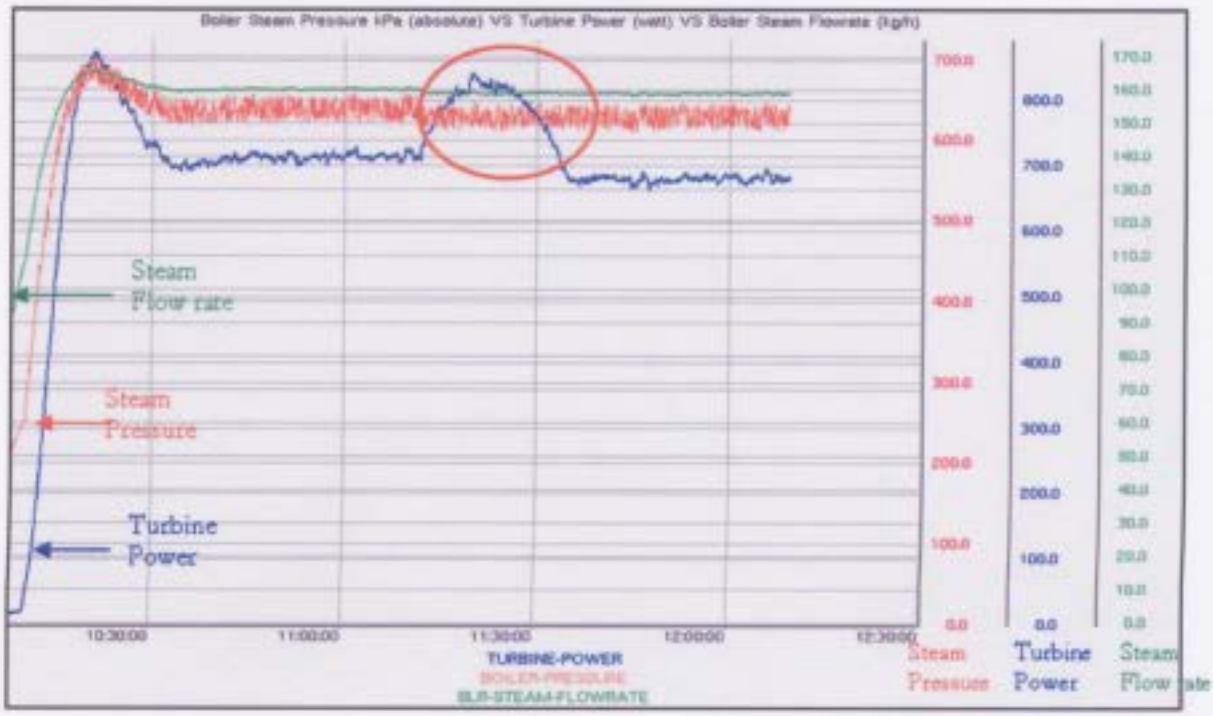


Figure 26 The simulated process outcome during non-daily operation

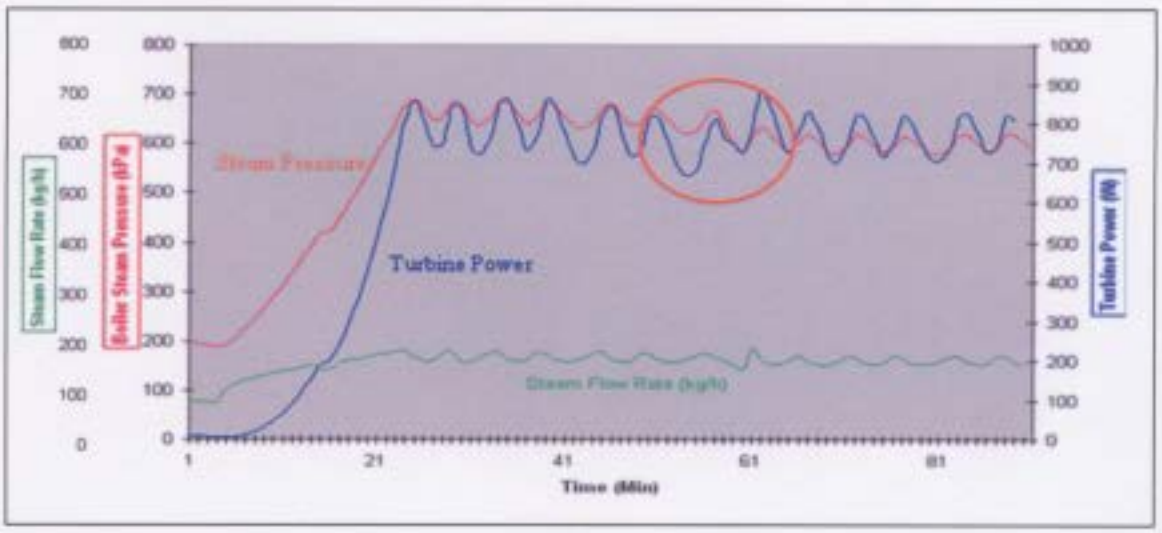


Figure 27 The unit trend record during non-daily operation

From the verification step, it has been proven that the developed model has the capability to represent the behaviors of the power unit by comparing outcome during normal and abnormal process conditions.

Summary

The developed process simulator is based on simplified process model and built by G2 structured procedural language. Since the accuracy and practicability of those model have been verified with the acceptable result using the past historical system data, this simulator is able to be used in the next stage.

Chapter 4

KNOWLEDGE-BASED REAL-TIME FAULT DIAGNOSIS METHOD

The simulator developed in last chapter provides a platform designing and testing the fault diagnosis function. The goal of this general function can be achieved by knowledge-based real-time fault diagnosis method, which is already introduced in section 2.2.2 and will be discussed further in this chapter. This method implements the valuable knowledge from the experts and operators as well as a vast databank of information from a variety of sensors. Fuzzy logic is used to make inferences according to the acquired information and knowledge. In addition, the development of a GDA-based computer application based on this method is also introduced in this chapter.

Already mentioned in section 2.2.2, the proposed method is comprised of three steps:

1. Acquiring information
2. Making inferences
3. Taking actions

4.1 Acquiring Information

Process data contains valuable information about the state, operation, and behavior of the process plant, more so in case with limited available process knowledge (Dash et al., 2004). In this step, a simple and quick method is needed to extract the meaningful information from the sheer volume of real-time sensor data. Process trend analysis is a useful approach to utilize real-time temporal patterns and it has previously been used in areas, such as process monitoring. The purpose of this step is to obtain real process data and thus perform process trend analysis.

4.1.1 Primitive Identification

The fundamental elements of trend description proposed by Janusz and Venkatasubramanian (1991) are primitives i.e., A(0,0), B(+,+), C(+,0), D(+,-), E(-,+), F(-,0), G(-,-) where signs are of the first and second derivative respectively (Dash et al., 2003) . A trend is represented as a sequence (combination) of these seven primitives (Dash et al., 2004). Figure 28 illustrates the shapes of these seven primitives. Clearly, identifying primitives is the first step of process trend analysis.

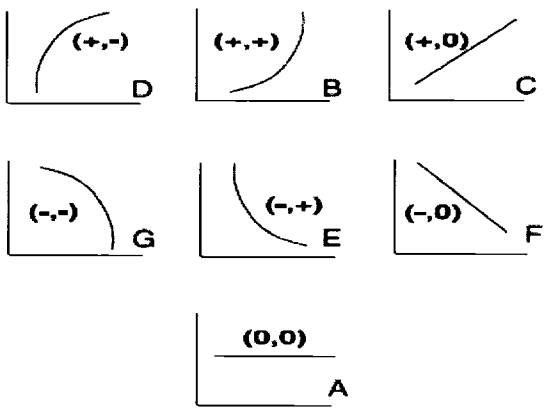


Figure 28 Fundamental elements of trend: Primitives (Dash et al., 2003)

Identifying primitives is not an easy task. There are several important issues which affect trend analysis and thus need to be discussed. These include noise, time scale, computational complexity and GDA feasibility:

- **Noise:** sensor data always contain noise. The quality and accuracy of primitive identification is affected by noise. The amount of noise must be minimized before performing any trend analysis.
- **Time Scale:** also referred as sampling rate in practical cases. The scale at which the primitive is extracted depends on the driving event. The time scale window should be wide enough to capture the significant variations.
- **Computational Complexity:** this is a difficult task. Depending upon the complexity of the process, the calculation might be extremely complicated. Since this

task requires real-time primitive extraction, computational complexity level should be considered as low as possible.

Taking the above points into consideration, a simple and efficient approach to identify real time primitives from raw sensor data is proposed in this step: *Fixed Window Discrete Data Primitive Identification.* The idea of this approach comes from the definition of the primitive and the characteristic of sensor information. The discrete sensor data is collected by the fixed window and fitted by third order polynomials. The instantaneous first discrete derivative (**FDD**) and second discrete derivative (**SDD**) are computed using general least squares fit method. The fixed window size is specified as five. The computation is based on the new sensor data value and four most recent data value, which is illustrated in Figure 29.

After obtaining both *FDD* and *SDD* using the method discussed above, current instantaneous primitive value can be identified according to the definition. For example, if *FDD* is 5.3 and *SDD* is -4.5, then primitive is considered as D.

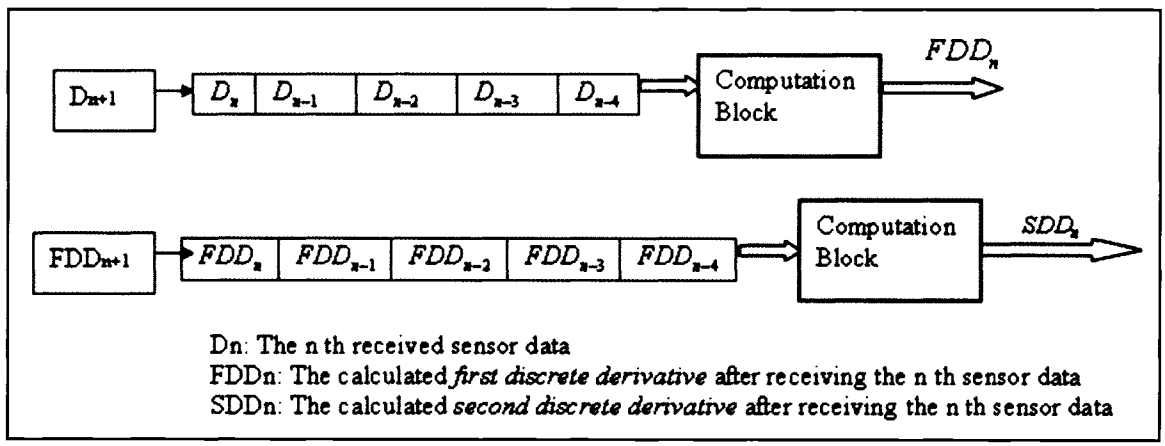


Figure 29 Fixed window discrete data primitive identification approach.

To illustrate this approach, simulated sinusoids signal data is generated as the input of primitive identification system. Figure 30 shows the result of this case study where the input signal and corresponding primitive output (horizontal line) are illustrated. In order to display

current primitive in the figure, the types of the primitives have been changed from characters (A - F) to numbers (1 - 7). Figure 31 shows a zoomed picture of the marked area.

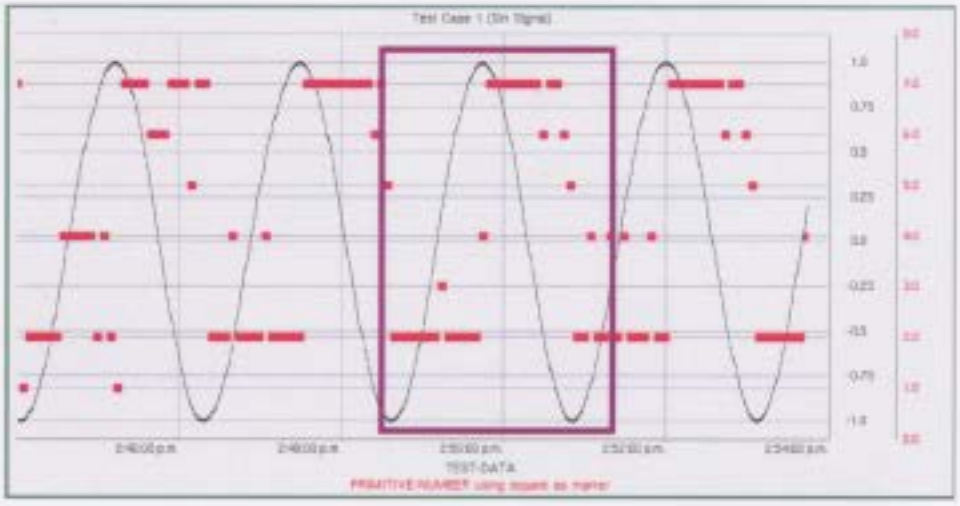


Figure 30 Result of primitive identification case study

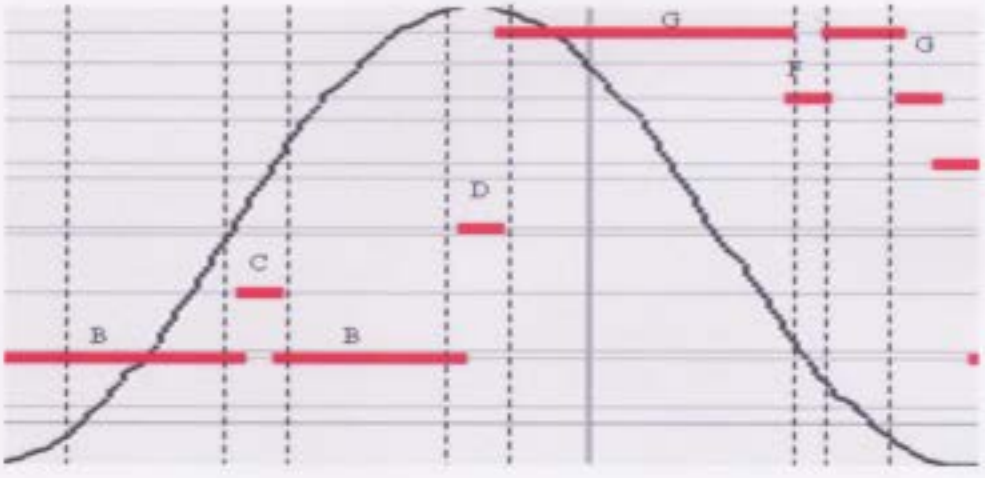


Figure 31 A zoomed view of dashed area in figure 29

Comparing the result from Figure 31 with primitive definition (Figure 28), it can be seen that the desired approach performs well in extracting basic primitives from the simulated input. Apparently, the advantage of this approach is that it is fast and efficient. However, it also has some limitations:

1) **dependency:** the output depends on the most recent five discrete input data. The quality of those data could affect the accuracy of identification. As mentioned before, this approach creates an instantaneous recognition and the result can not be changed later.
2) **noise tolerance:** noise is still a major issue. There is no noise canceling technique used in this approach. The input sensor data need to be filtered before performing any analysis.

These limitations can be minimized to some extent by modifying sampling rate (window size) and creating filters in GDA application. Since the proposed fault diagnosis methodology is based on real time data and needs a rapid response, this may be considered as a satisfactory response.

4.1.2 Process Trend

As mentioned earlier, a trend is the combination of seven primitives:

Trend Tr= $\{P_1, P_2, \ldots\ldots P_N\}$

For example, Figure 31 shows a trend composed of three primitives BDE, which can be represented as Tr= {B, D, E}

Figure 32 An example of BDE trend

The purpose of using process trend is to capture the pattern of fault event for future analysis. Therefore, the comparison between two trends is necessary. However, due to the uncertain characteristic of primitive identification and the similar shape between some primitives, it is

not practical to perform a strict comparison. For example, in Figure 33, Trend DG and CG are similar to some extent, since the shape of primitive D and primitive C are alike.

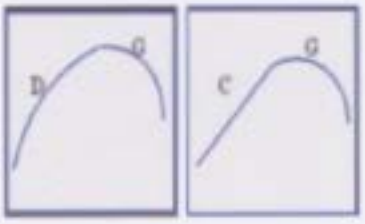


Figure 33 Example of trend DG and CG

In order to solve this issue, Similarity Index (**SI**) (Dash et al., 2003) is introduced to represent the similarity level between two trends. Table 3 shows the pre-defined similarity matrix between each primitive, where $S_{P_i P_i^*}$ provides the similarity between P_i and P_i^* (from 0 to 1). For example, D and C are more similar than D and E, given that S_{DC} (0.75) is larger than S_{DE} (0). The SI between two trends can be calculated by the equation (1).

$$SI = \frac{\sum_{i=1}^{N} Sp_i p_i^*}{N} \qquad (1)$$

It should also be noted that the similarity matrix in Table 3 is not fixed. The developers can improve the accuracy of the calculated SI by adjusting each related $Sp_i p_i^*$.

Table 3 Primitive similarity matrix

$Sp_i p_i^*$	A	B	C	D	E	F	G
A	1	0	0.25	0	0	0.25	0
B	0	1	0.75	0.5	0	0	0
C	0.25	0.75	1	0.75	0	0	0
D	0	0.5	0.75	1	0	0	0
E	0	0	0	0	1	0.75	0.5
F	0.25	0	0	0	0.75	1	0.75
G	0	0	0	0	0.5	0.75	1

Figure 34 illustrates the algorithm of computing SI after receiving the identified primitive. Initially, knowledge-based trend Tr^* must be determined, which includes the number and type of primitives. Then similarity value (S_{pp_i}) is decided after comparing each received primitive with corresponding knowledge-based primitive in Tr^*. If S_{pp_i} is not equal to zero, the current SI is calculated. The SI computation ends when either index is equal to N or the next similarity value is zero.

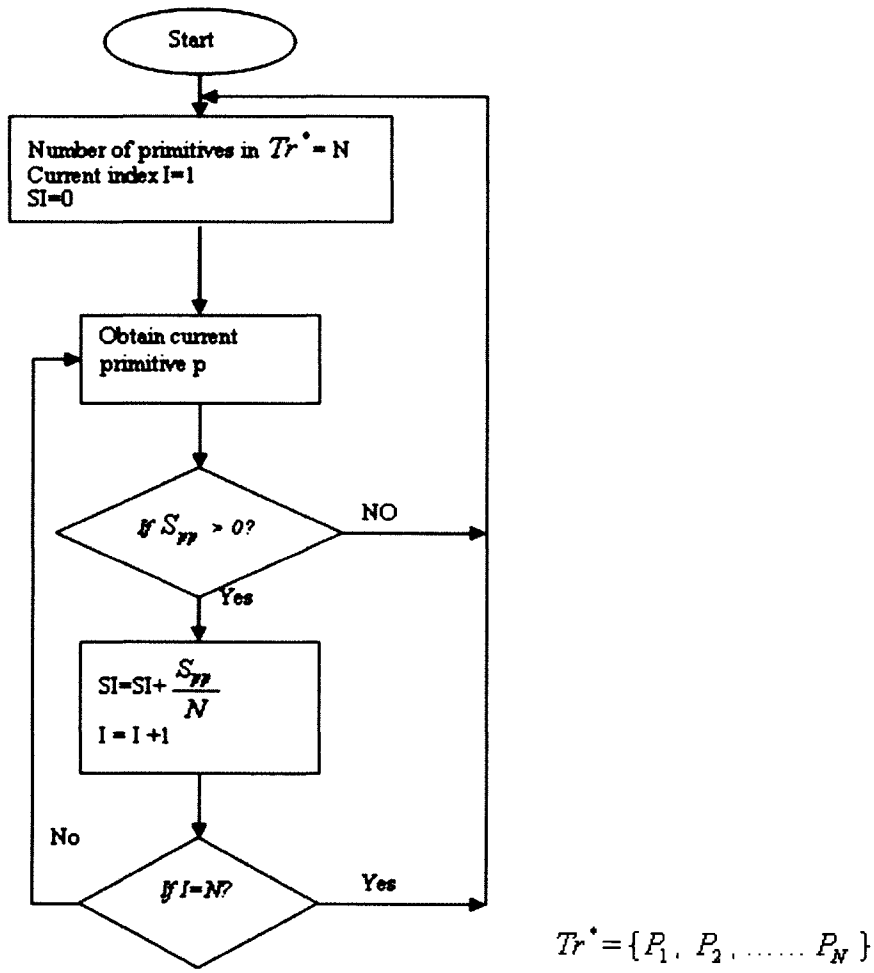


Figure 34 The algorithm of computing SI

The Rate of Change (**ROC**), which is the first derivative of corresponding process variable, represents the discrete rate of change. It is obtained through computing the instantaneous

slope for five individual input data using general least squares fit method. ROC can be used to characterize the input sensor data by determining whether and at what rate the input is increasing or decreasing. Comparing with SI, ROC is capable of quantifying the temporal pattern of sensor data. Therefore, it may also be considered as input to the analysis.

4.2 Making Inferences

According to the previous section, the inputs of the Inference system are SI (similarity index) and ROC, obtained from process trend analysis. One important issue here is that the SI is not a precise input. It is difficult to develop a model-based inference system for the purpose of obtaining a precise output. Another issue is that it is still hard to utilize the useful information provided by engineers to the inference system. Motivated by the issues mentioned above, a *fuzzy inference system* (**FIS**) is proposed in this step.

4.2.1 Fuzzy Inference System

Fuzzy inference system can be considered as an inference system based on both expert knowledge and fuzzy logic. FIS has the capability of converting the numeric data into linguistic variables. The imprecision and uncertainty characteristic of system inputs is managed using fuzzy sets. Furthermore, using FIS approach to perform fault diagnosis is also able to handle the impreciseness of process trend representation. It is worth mentioning that fuzzy inference systems have been successfully applied in fields such as automatic control, data classification, and decision analysis (Marcellus, 1997). Below a brief explanation of fuzzy logic and fuzzy set is provided, and for details, please refer to a paper by Dr. Zadeh (Zadeh, 1988).

A fuzzy logic system is a nonlinear system whose behavior is described by a set of linguistic rules. For example, rules such as:

IF (service is good) THEN (give more tips)
IF (service is alright) THEN (give average tips)
IF (service is bad) THEN (give less tips)

Unlike other regular mathematical systems, fuzzy logic system is related to the classes with unsharp boundaries where the output is only the matter of degrees. It is primarily about linguistic vagueness through its ability to allow an element to be a partial member of set, so that its membership value can lie between 0 and 1(Harris et al., 2002).

Central to the fuzzy logic system are fuzzy sets and membership function. A fuzzy set A is defined as a set of ordered pairs:

$$A = \{(x, \mu_A(x)) \mid x \in X\}$$

where A is called the fuzzy sets and $\mu_A(x)$ is called the membership function.

Fuzzy logic is conceptually easy to understand and flexible. Imprecise data can be easily processed by it. Fuzzy logic is based on the natural language and convenient to use in the complicated control system.

4.2.2 Application of FIS to Fault Diagnosis

To identify the fault, expert knowledge is mapped with the knowledge-based fault process trend (pattern) in the form of fuzzy if-then rules. An *If-then* rule typically expresses an inference such that, if we know a fact, then we infer or derive, another fact called a conclusion (El-Shal & Morris, 2000). For example a rule might read,

If sensor S1 shows Tr1 AND ROC of sensor S1 is large, then the fault F1 is most likely to happen

This rule implies that if sensor S1 has been observed with process trend Tr1 and at the same time its value increases significantly, then the possibility of F1 fault event occurring is extremely high. In this example, Tr1 is knowledge-based process trend, which has been recognized as a fact by the experts based on their experiences. To evaluate this rule, current *SI* is measured using the algorithm introduced in the previous section and fuzzificated by corresponding input membership function. The detailed demonstration of this step is presented through a case study later.

4.3 Taking Actions

The objective of this step is to guide the process back to normal in the case of abnormal conditions. After detecting an abnormal event, which could cause severe accidents, the proper actions are required immediately. This will be achieved by developing a set of actions which include activating safety measures and a higher layer of protection.

4.4 Developing GDA Application

In subsequent section, details of the implementation of methods and approaches discussed above using GDA, which is introduced in section 2.2.3, are provided.

4.4.1 Primitive Identification

Figure 35 illustrates the snapshot of the GDA application for the purpose of primitive identification. This application consists of several blocks including:

- **Real-time Process Data:** this block is the source of information. It could be industrial data such as the data obtained from the sensor or the simulated data. The output frequency of this block should be high enough in order to capture the fast updated pattern.
- **Filter:** *non-linear exponential filter* is added in order to filter out high frequency noise. A non-linear exponential filter is a low pass filter, which is able to filter high frequency noise. The advantage of this filer is that it can filter noisy input but also be able to respond to the significant changes quickly. Furthermore it also improves the accuracy of primitive identification.
- **Discrete rate of change:** this block is responsible for computing the instantaneous rate of change for its input using a general least squares fit method. In order to apply proposed *Fixed Window Discrete Data Primitive Identification* approach into this block, window size is set to 5 and polynomial order is set to 3.
- **Logic gates blocks:** the aim of these blocks is to determine the primitive type according to the value of *FDD* and *SDD*.

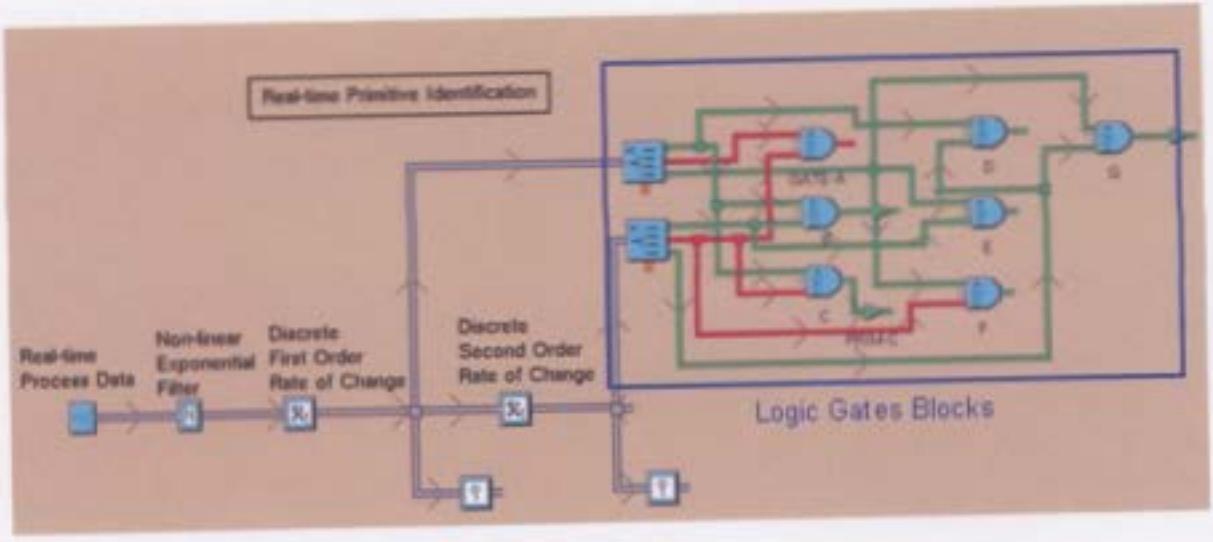


Figure 35 snap shot of primitive identification application

4.4.2 *Similarity Index Computation*

The similarity index computation follows the algorithm discussed in previous section. Figure 36 shows the snapshot of this application, which includes two components: logic gates blocks and G2 procedure program.

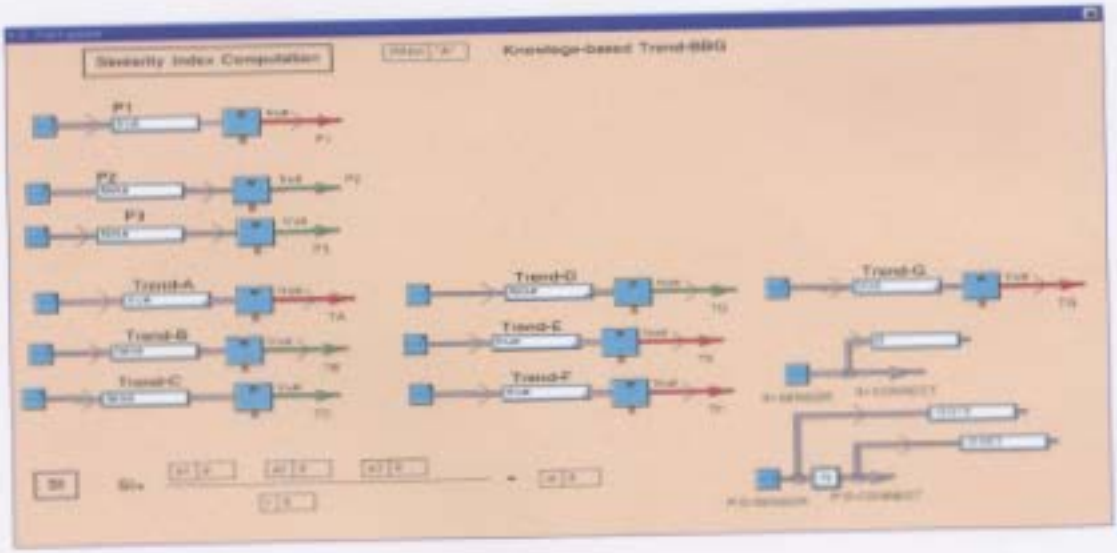


Figure 36 The snapshot of SI computation application

4.4.3 *Fuzzy Inference System (FIS) Implementation*

GDA supports fuzzy logic implementation. There are several blocks specially designed for developing fuzzy logic components such as Fuzzy Consequence Block, Weighted Evidence Combiner Block, and Fuzzy Evidence Gate Block. However, it should be noted that one fuzzy evidence gate block only enables three combined *if-then* rules.

Figure 37 shows a snapshot of the developed fuzzy logic system using GDA blocks. It also illustrates various parts of a fuzzy logic system with related GDA block used to implement that specific part. As mentioned above, only three rules are allowed. More rules can be implemented by adding similar applications. However each output should be combined for the final output, which is illustrated in Figure 38.

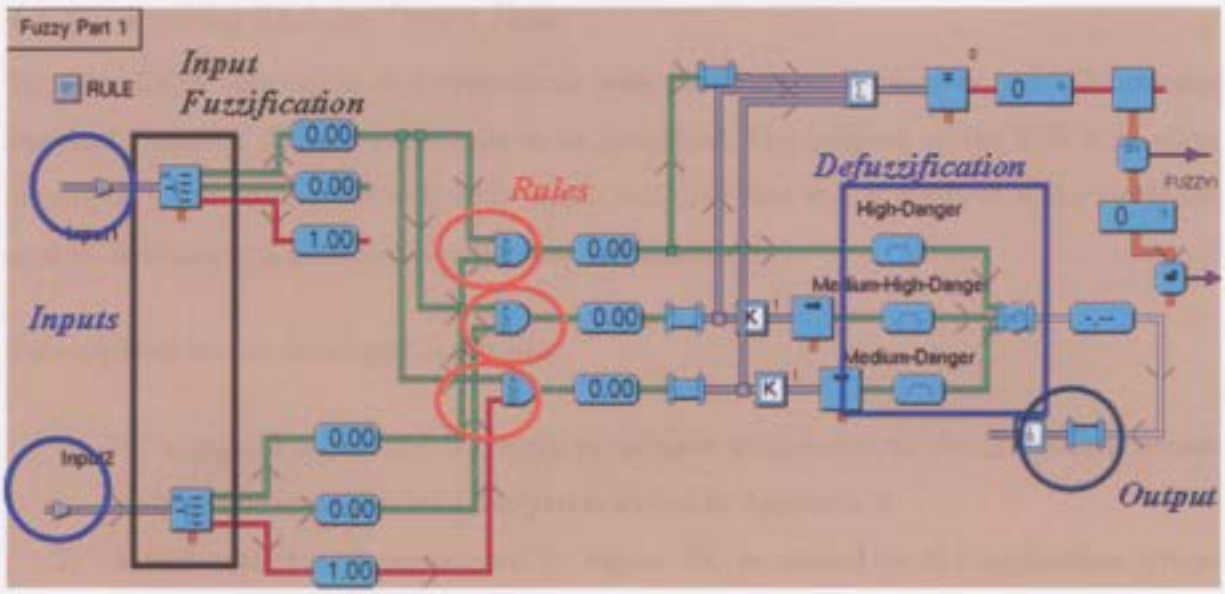


Figure 37 The snapshot of GDA application of fuzzy logic system

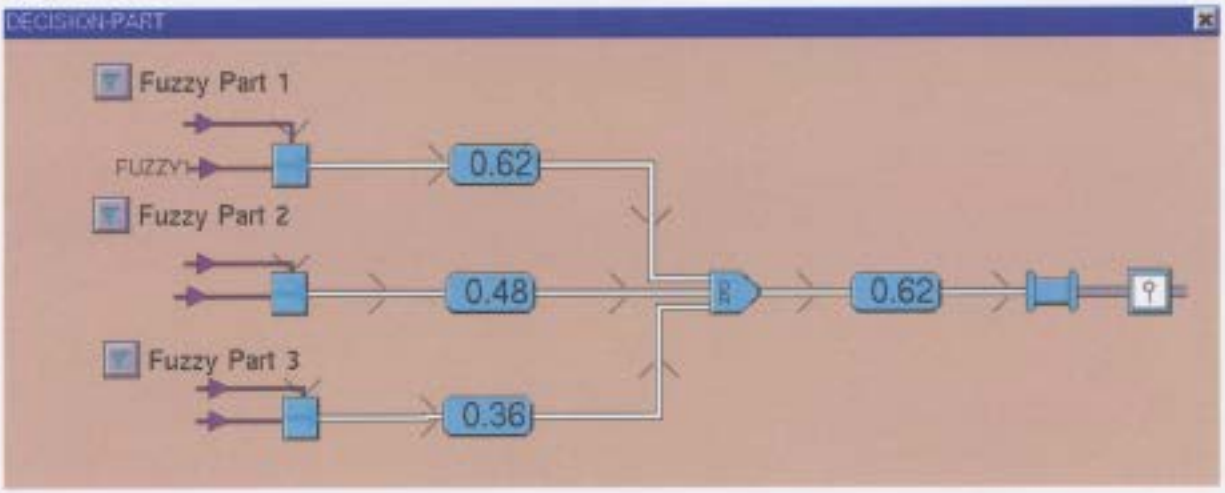


Figure 38 The snapshot of GDA application of fuzzy logic final output

4.4.4 *Acquiring Real-time Process Data*

To allow GDA application to communicate with external process sensors, the G2 Gateway Standard Interface (**GSI**) object needs to be generated. The purpose of the GSI is to allow developed GDA application to quickly obtain real-time data that it needs to make intelligent control decisions in a time-critical processing environment.

Two applications are developed in this part:

1) "C" compiled application is created in monitor workstation to obtain real-time sensor reading. The source code of this part is shown in Appendix B.
2) A GSI interface, demonstrated in Figure 39, is created in G2 application whose objective is to get value of desired variable from "C" compiled application.

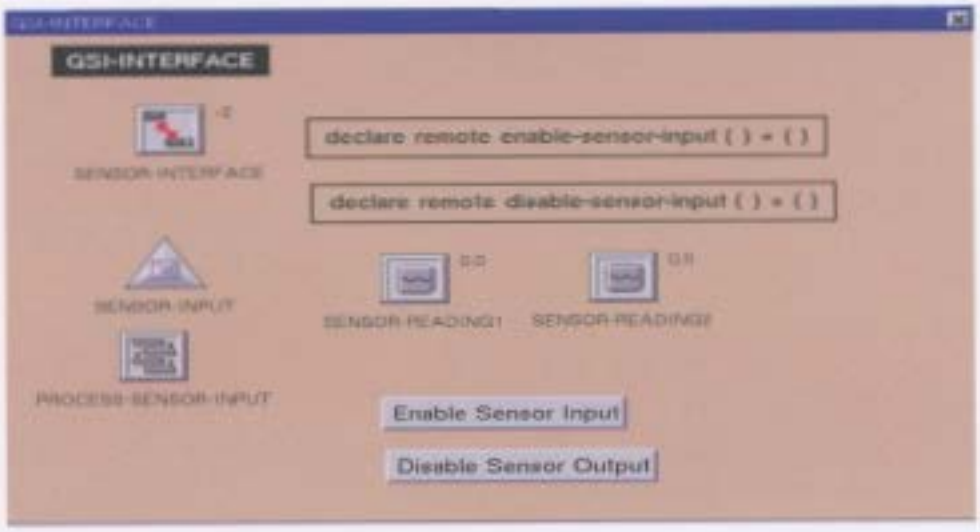


Figure 39 The snapshot of a GSI interface

Summary

To develop general fault diagnosis function, this chapter proposes a knowledge-based real-time *fault diagnosis method. It is a flexible and efficient method to reason about system behavior,* more so when system model is complex and difficult to build. In addition, a G2-based computer tool is developed in order to realize this method.

Chapter 5

APPLICATION OF METHODOLOGY AND G2 BASED TOOL

In order to verify the capability and efficiency of the proposed methodology, two case studies are presented in this chapter. The developed GDA application with fault diagnosis function is demonstrated on a variety of situations. The first case study is carried out to detect well-defined fault event in a micro steam power unit using simulated data. Second case study is conducted to identify abnormal material temperature drop using real-time monitored industrial data.

5.1 Case Study 1: Fault Event Detection in a Micro Steam Power Unit Using Simulated Data

In this case study, the input data is generated by the developed micro steam power unit simulator, which is discussed in chapter 3. The purpose of this case study is to test the performance of the developed GDA application before applying it into real process system. The first step is to determine the knowledge-based fault event.

5.1.1 *Fault Event Determination*

After taking suggestions from both the expert and demonstrator of this power unit, the fault event (**F1**) that most likely will lead the whole system to unsafe state has been identified with two patterns:

1) The trend pattern of steam pressure in the boiler during this specific event can be recognized as ***BBG*** (**P1**)
2) Steam pressure suddenly increases or decreases significantly(**P2**)

The steam pressure in the boiler can be obtained from sensor sp-1.

5.1.2 *Fuzzy Inference System*

According to the characteristics and patterns of knowledge-based fault event F1, the inputs and fuzzy logic system have been modified, which are shown below:

Inputs:

1) SI of sp-1
2) Absolute ROC of sp-1*(|ROC|)*

Output:

The possibility of this event happening: *high, medium high, medium, medium low, low.*

Membership Function

The membership function (**MF**) essentially embodies all fuzziness for a particular fuzzy set (El-Shal & Morris, 2000). The shape of membership functions used for both input and output are either triangular or trapezoidal.

1) Input Membership Function:

Three membership functions are selected for both inputs, with linguistic values: low, medium, and high. The range for each MF is shown in Table 4 and MF graph is shown in Figure 40.

Table 4 range of MFs for both inputs

Input	Low	Medium	High
SI	<0.4	> 0 and <0.9	>0.5
|ROC|	<2.5	>1 and <8.25	>8.25

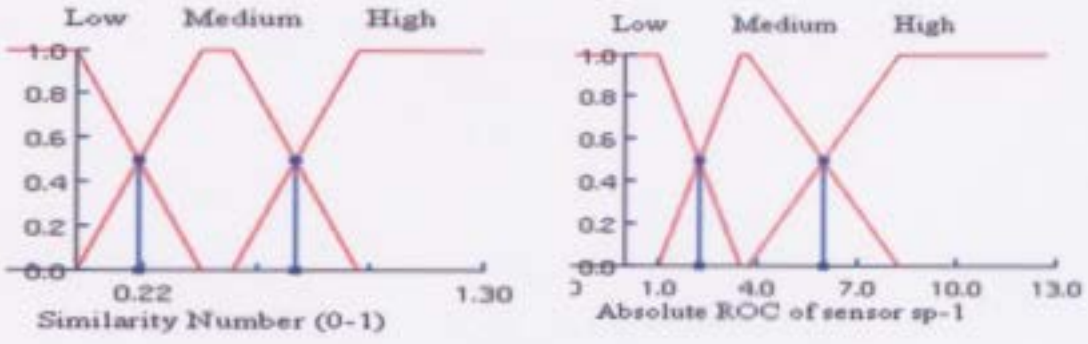


Figure 40 Membership function graph of both inputs

2) Consequence Membership Function:

Five output (consequence) functions are selected. The purpose of these functions is to determine the likelihood of the conclusion which is true, given a premise. The range for each MF is shown in Table 5 and MF graph is shown in Figure 41.

Table 5 The range of MFs for output

Level	Low	Medium Low	Medium	Medium High	High
Consequence	<0.2	> 0 and <0.475	>0.25 and <0.85	> 0.5 and < 0.7	>0.73

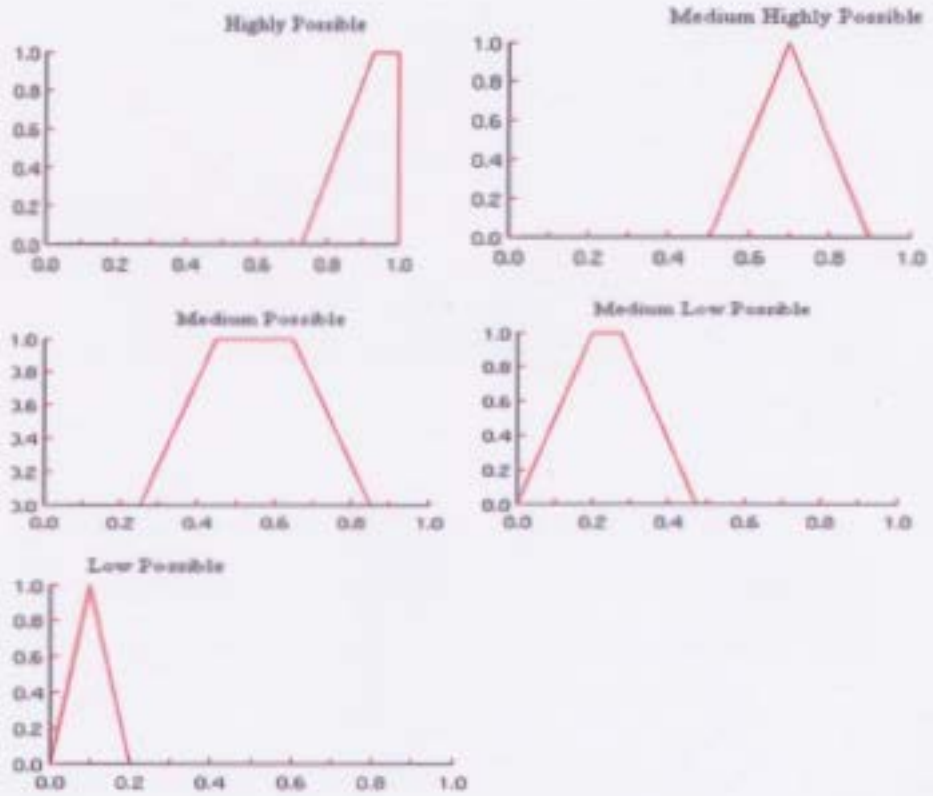


Figure 41 The graph of consequence membership functions

Rules:

Table 6 displays all fuzzy decision-making rules derived from knowledge base, which are developed using operator's experiences. For example the rule in the circle can be read as:

If *SI of sensor SP-1 is medium* ***AND*** *absolute value of ROC of sensor SP-1 is medium,* ***then*** *the possible happening level of F1 event is medium.*

Table 6 Rule table

Possibility Level		**SI of Sensor SP-1**		
		High	*Medium*	*Low*
Absolute ROC Of Sensor SP-1	*High*	High	Medium High	Medium
	Medium	Medium High	Medium	Medium Low
	Low	Medium	Medium Low	Low

Center of Gravity defuzzification method is implemented for combining all the consequences in order to make decision, which is illustrated in Equation 2. Basically this method calculates the weighted average of the center values of the consequence membership function centers.

$$u^{crisp} = \frac{\sum_i b_i \int \mu_{(i)}}{\sum_i \int \mu_{(i)}} \qquad (2)$$

where b_i denotes the center of consequence membership function

μ_i denotes the membership function

5.1.3 *Diagnosis Testing Result*

In order to evaluate the proposed methodology, the micro steam power unit simulator is activated under normal process conditions. The first step of the diagnosis test is to verify the performance of FIS. It is very important to confirm that FIS is capable to identify the pre-defined patterns of fault event successfully.

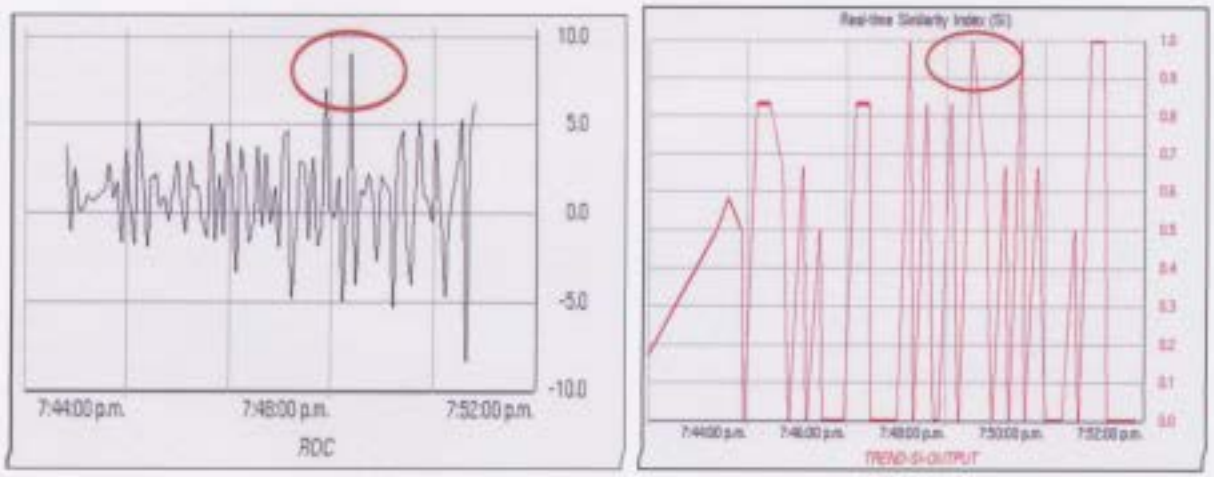


Figure 42 The ROC and SI output of boiler steam pressure

Figure 42 illustrates the observed FIS inputs, which are real-time ROC and SI of boiler steam pressure. Although ROC value oscillates between -5.0 and 5.0, it is easy to see that the significant change has occurred at some point. In the marked circle area, ROC has reached to

12, which is higher than average and SI value is close to 1 at the same time. According to the design of FIS, its outcome should also reach the maximum consequently.

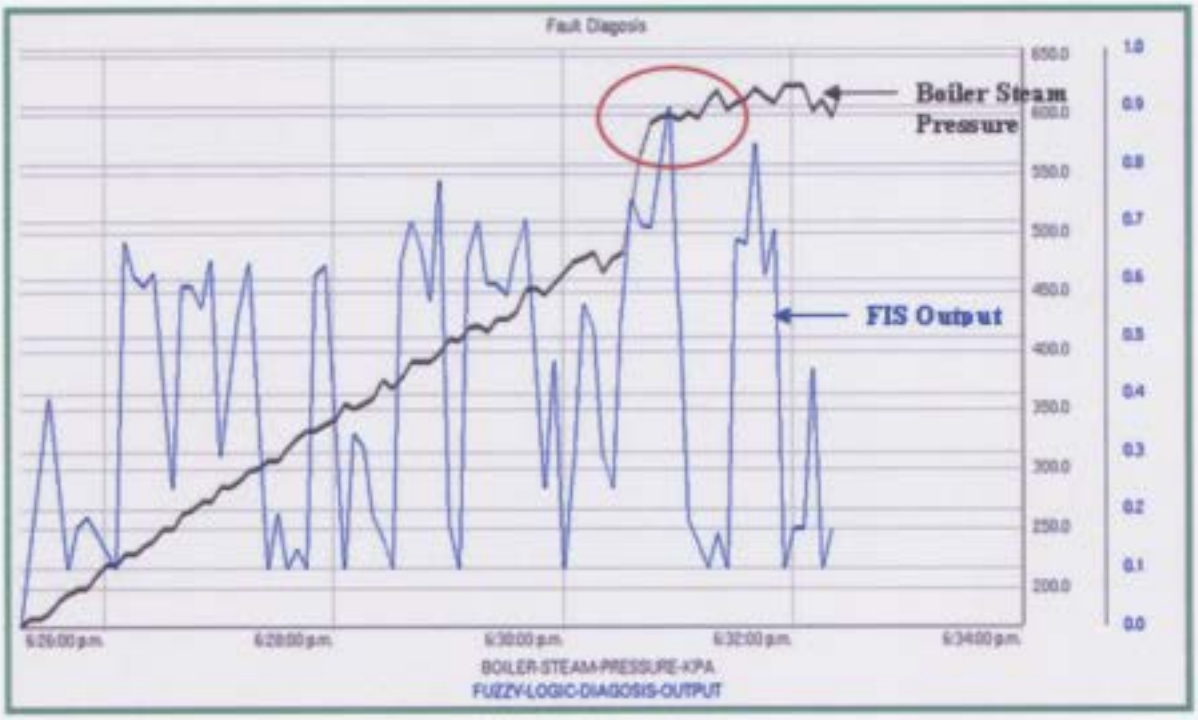


Figure 43 Fuzzy logic diagnosis output (test)

Figure 43 illustrates the outcome of simulated boiler steam pressure and FIS output. It can be seen that FIS output oscillates between 0.1 and 0.8 normally except in the marked circle where output has reached 0.91. This is due to the significant change in two FIS inputs described above (Figure 42). Therefore, the observed FIS output is the same as expected. However, this only proves that FIS output is reasonable and can be explained by pre-defined knowledge. It is not enough to raise any alarm. In order to improve the accuracy of developed FIS application and fulfill the purpose of triggering warnings in case of critical operations, the conditions that will make raise-alarm decision should be determined. For example, the corresponding intelligent alarm will be raised only after detecting three consecutive outputs that exceed pre-defined threshold value. After studying the performance of developed application and discussing with the system operator, FIS output threshold is set to 0.85 and the number of recurring outputs beyond threshold in 3 minutes is set to 3.

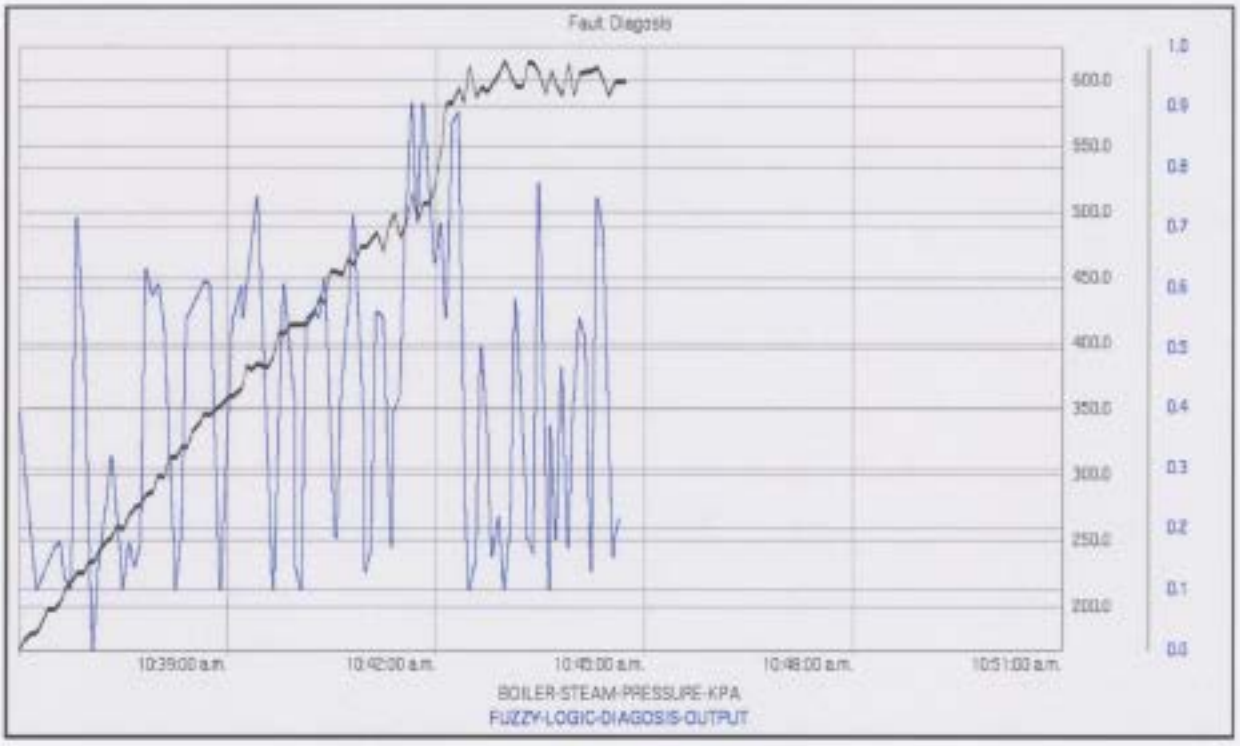


Figure 44 Fuzzy logic diagnosis output

Figure 44 illustrates the FIS diagnosis output after adding an intelligent alarm to the simulator. As observed from this figure, boiler steam pressure starts to rise from 200 kPa in the beginning. When it rises to around 500 kPa, fault event F1 is generated and boiler steam pressure will close to critical condition in a couple of minutes after that. Figure 45 shows a zoomed marked area of Figure 44.

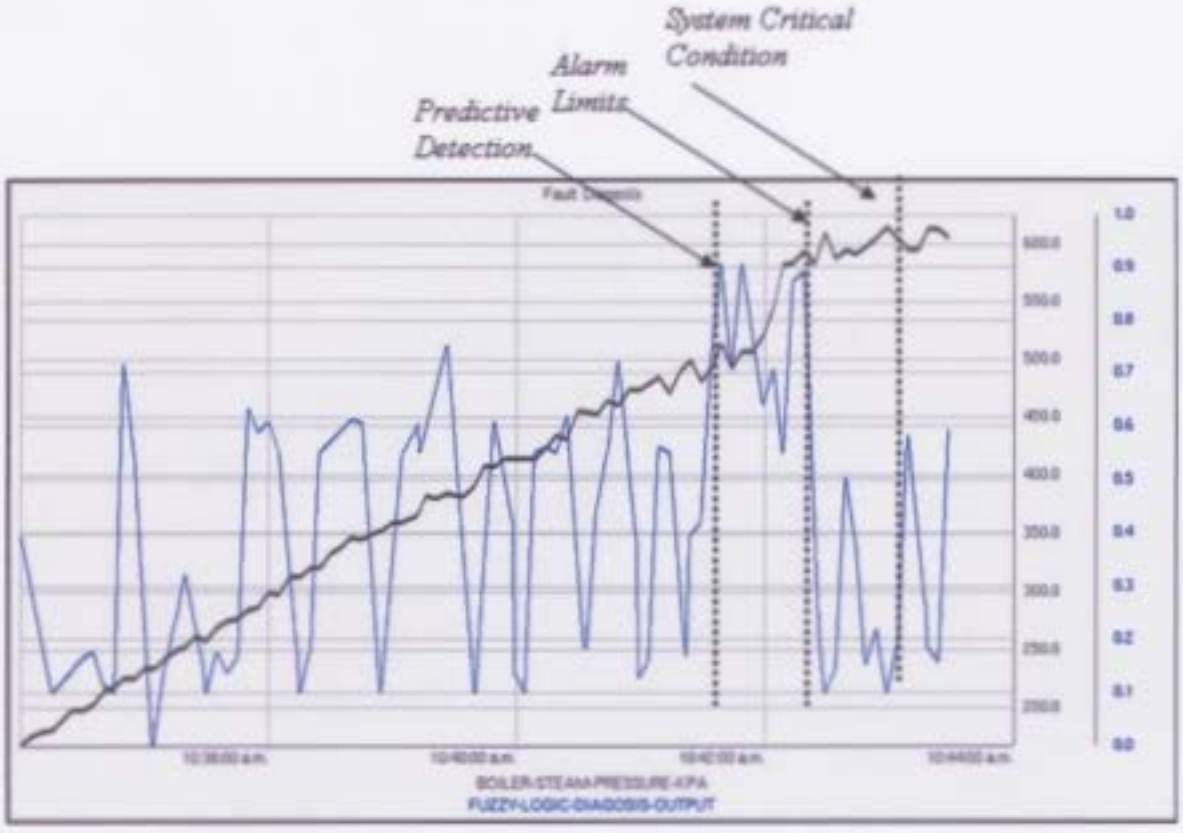


Figure 45 Zoomed marked area of Figure 44

It can be seen from Figure 45 that as boiler steam pressure rises, at some points fault event will first be detected (predictive detection) since FIS output exceeds the threshold at first time. After that, this situation repeats twice. So the observed FIS output satisfies the conditions to make raise-alarm decision. As a result, the intelligent alarm will be raised in order to notify the operator prior to the boiler steam pressure reaching critical operating condition (when steam pressure reaches around 625 kPa).

The observed FIS output can be explained by retrieving earlier developed knowledge. This provides clear evidence that the proposed methodology has the capability of detecting the defined fault event F1 effectively.

The developed GDA application is only designed for detecting the fault event F1. However, it can be modified for detecting other fault events as well. For example, if a fault event has been

identified with three patterns, then the number of fuzzy logic input should be three. Furthermore, all membership functions have to be adjusted in order to obtain accurate and reasonable results.

5.2 Case Study 2: Abnormal Material Temperature Drop (real-time industrial data)

In order to test the developed GDA application in real process system, three fluid chemical samples are heated under high pressure. The temperature is manipulated by a PID controller. When temperature reaches around 100 degrees Celsius, the heater will be suddenly shut down. Due to this unexpected operation, the temperature of heated samples will start to reach beyond the normal operating level. This abnormal condition might not be detected by a conventional control system and difficult for an operator to identify. In order to test the feasibility of the proposed methodology in a real industry situation, this event is performed using *Advanced Reactive System Screening Tool* (**ARSST™**) from Fauske & Associates, LLC.

Figure 46 *The picture of ARSST standard containment vessel*

ARSST™ system, shown in Figure 46, is the product of extensive research into runaway chemical reactions and their impact on process system dynamics. The ARSST™ is designed to simplify the acquisition of data necessary for thermal hazards analysis, runaway reaction evaluations, and the proper sizing of pressure relief vents (*http://www.fauske.com/arsst.asp*). This system is located in the Health, Safety and Risk Engineering Lab in the Inco Innovation Centre of Memorial University and is used for the purpose of graduate student experiments. The detailed procedure to perform this test in ARSST™ system is included in Appendix E.

Three chemical samples, which are used in this case study, are H_2O, sodium hydroxide solution (NaOH, 1.60 mg/ml) and methyl red ($C_{15}H_{15}N_3O_2$ 0.1%).

5.2.1 *Fault Event Determination*

After observing this abnormal thermal phenomenon, the fault event (**F2**) that most likely will lead to it has been identified with two patterns:

1) The trend pattern of sample temperature during this specific event can be recognized as ***GGB*** (**P3**)
2) Sample temperature suddenly decreases significantly(**P4**)

The sample temperature can be obtained from thermocouple TC-1.

5.2.2 *Fuzzy Inference System*

According to the characteristics and patterns of knowledge-based fault event F2, the inputs and fuzzy logic system have been modified, which are shown below:

Inputs:

1) SI of TC-1
2) *Absolute ROC of TC-1*

Output:

The possibility of this event happening: *high, medium high, medium, medium low, low.*

Membership Function:

The shape of membership functions used for both input and output are either triangular or trapezoidal.

1) Input Membership Function:

Three membership functions are selected for both inputs in each test case, with linguistic values: low, medium, and high. The range for each MF and MF graph are shown below:

Table 7 The range of MFs for both inputs for sample H2O

Input	Low	Medium	High
SI	<0.4	> 0 and <0.9	>0.5
|ROC|	<0.04	>0.01 and <0.06	>0.04

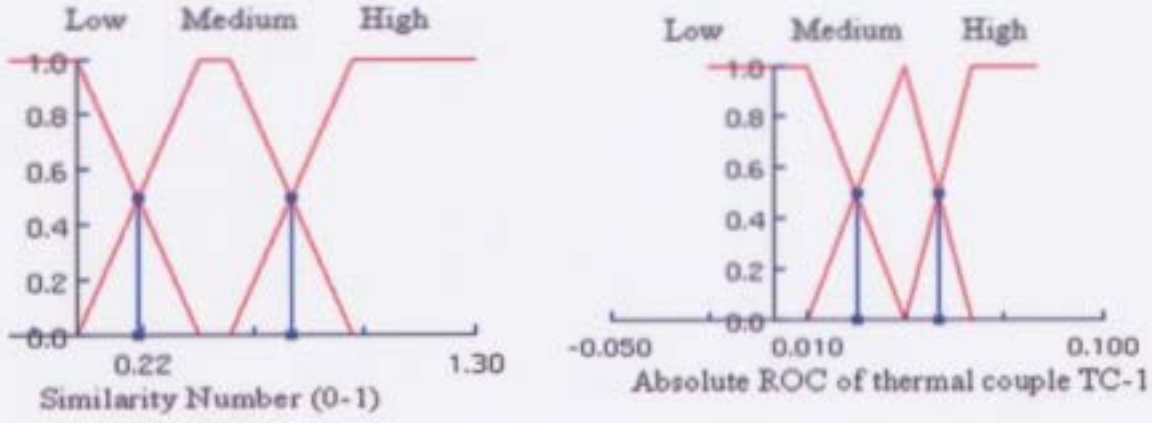


Figure 47 function graph of both input for sample 1 (H_2O)

Table 8 The range of MFs for both inputs for sample 2 (sodium hydroxide)

Input	Low	Medium	High
SI	<0.4	> 0 and <0.9	>0.5
|ROC|	<0.06	>0.01 and <0.1	>0.06

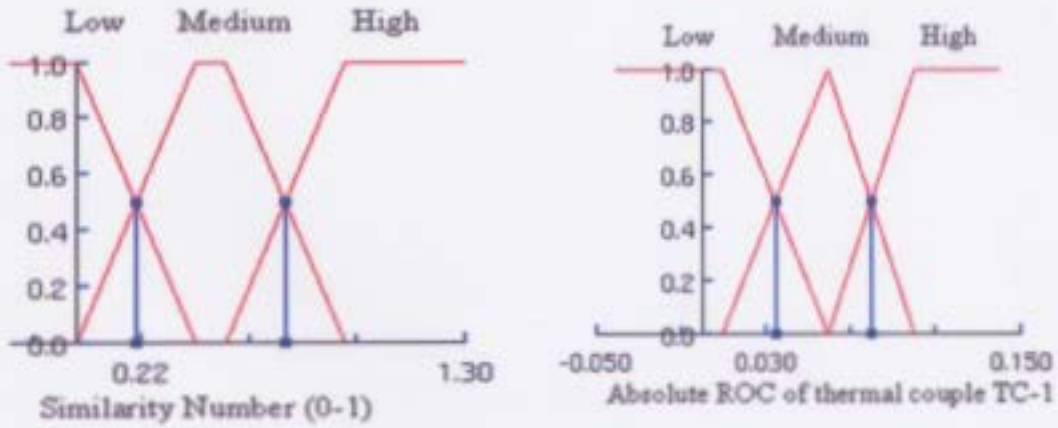


Figure 48 Membership function graph of both input for sample 2 (sodium hydroxide)

Table 9 The range of MF's for both inputs for sample 3 (methyl red)

Input	Low	Medium	High
SI	<0.4	> 0 and <0.9	>0.5
|ROC|	<0.05	>0.01 and <0.07	>0.05

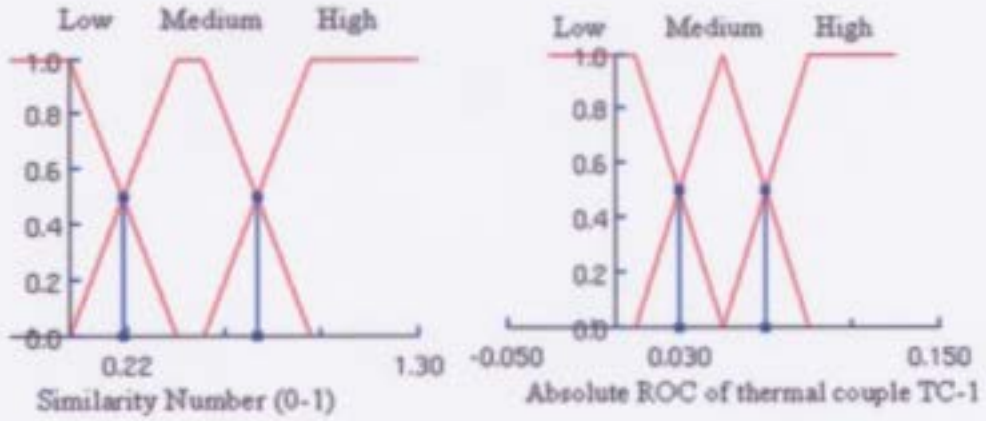


Figure 49 function graph of both input for sample 3 (methyl red)

2) Consequence Membership Function:

The range for each MF is shown in Table 10 and MF graph is shown in Figure 50.

Table 10 The range of MF's for output

Level	Low	Medium Low	Medium	Medium High	High
Consequence	<0.2	> 0 and <0.475	>0.25 and <0.85	> 0.5 and < 0.9	>0.73

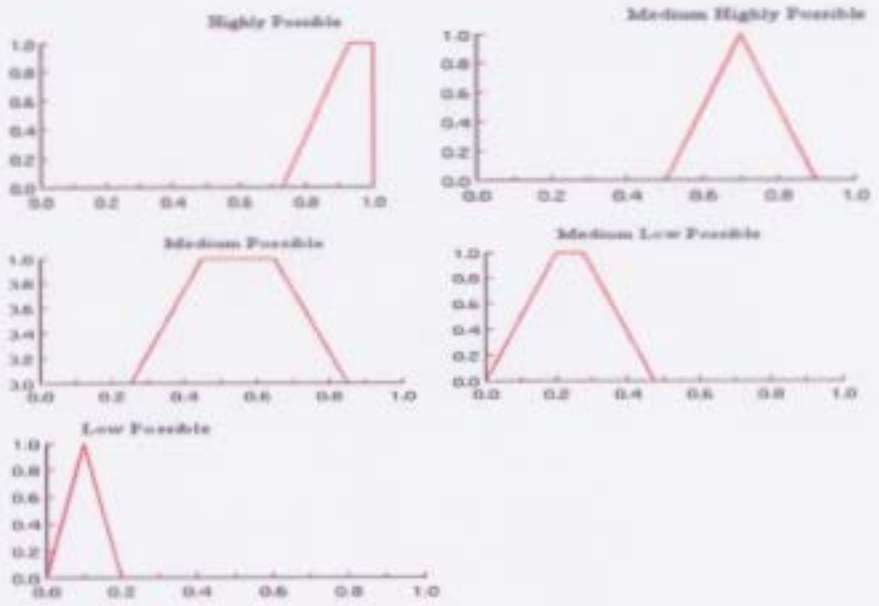


Figure 50 The graph of consequence membership functions

Rules:

Table 11 displays all fuzzy decision-making rules derived from knowledge base for this case study.

Table 11 Rule Table for case study 2

Possibility Level		**SI of Sensor TC-1**		
		High	*Medium*	*Low*
Absolute ROC Of Sensor TC-1	*High*	High	Medium High	Medium Low
	Medium	Medium High	Medium	Medium Low
	Low	Medium	Medium Low	Low

5.2.3 *Diagnosis Testing Result*

Three chemical samples are heated at a rate of 2 degrees Celsius per minute under pressure of 120 *Psi* using ARSST containment vessel. The output of thermocouple TC-1 is obtained through a DAS (*Data Acquisition*) card installed on the system monitor workstation. In order

to generate fault event F2, the heater is turned off when sample temperature reached around 100 degrees Celsius.

The aim of this test case is the same as the previous test case. The only difference is that the inputs are generated by a real process system, not by simulation.

As discussed in the previous case study, a similar intelligent alarm is added to the application. In addition, the conditions that will make raise-alarm decision are also determined as follows:

1) FIS output threshold value is set to 0.88
2) The number of recurring output beyond threshold is set to 4

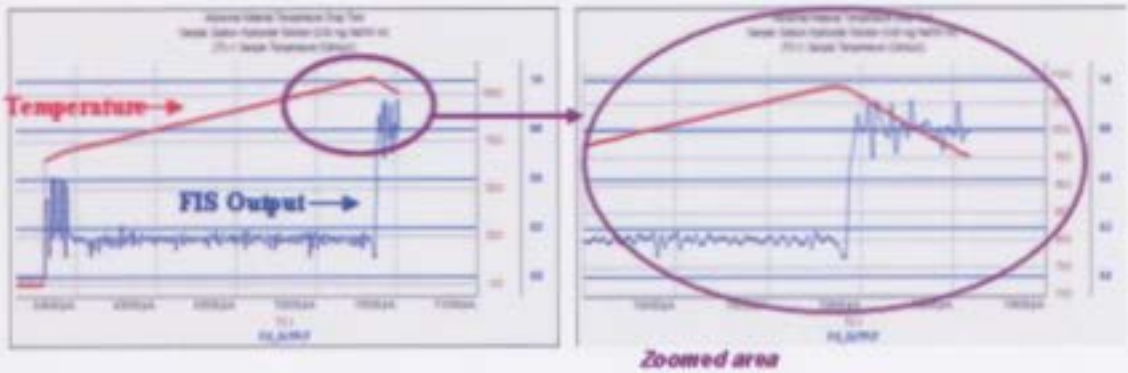
Figure 51 to 53 illustrate the outcome of sample temperature and FIS output for three tested materials respectively. To compare the output of normal and abnormal temperature drop, the outcome of system during normal operation is also illustrated.

As observed from those figures that the FIS output oscillates between 0.1 and 0.5 normally, except in the marked circle where it has exceeded threshold value. Sample temperature starts to rise in the beginning. When it reaches around 100 degrees Celsius, fault event F2 is generated. The temperature begins to drop and will reach critical operating condition in several minutes after that.

Final results from this test case are similar to test case 1. As sample temperature drops, at some points fault will first be detected since FIS output exceeds the threshold at first time. After that, this situation repeats three times. Therefore, the observed FIS output satisfies the condition to make a raise-alarm decision. As a result, warning alarm will be raised in order to notify the operator prior to the system reaching critical condition.

Compared with the abnormal situation, the maximum value of corresponding normal situation output is always below 0.6, which is far below threshold value. This comparison means the developed FIS has the capability of identifying the variation between normal and abnormal condition.

Sodium hydroxide solution *fuzzy logic diagnosis output (Abnormal temperature drop)*



Sodium hydroxide solution *fuzzy logic diagnosis output (Normal temperature drop)*

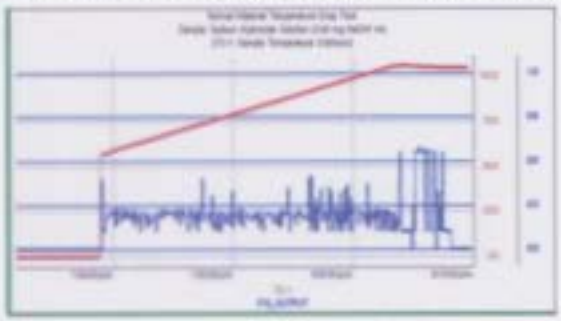
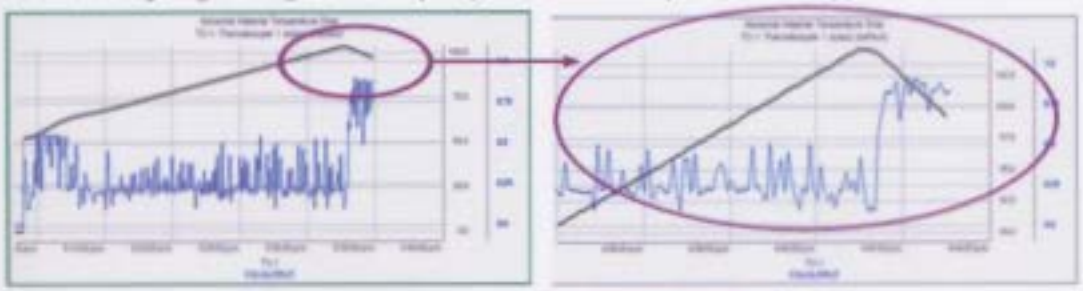


Figure 51 Sample 1: sodium hydroxide solution (NaOH)

Water *fuzzy logic diagnosis output (Abnormal temperature drop)*



Water *fuzzy logic diagnosis output (Normal temperature drop)*

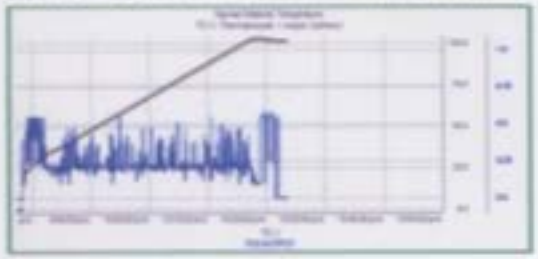


Figure 52 Sample 1: water (H2O)

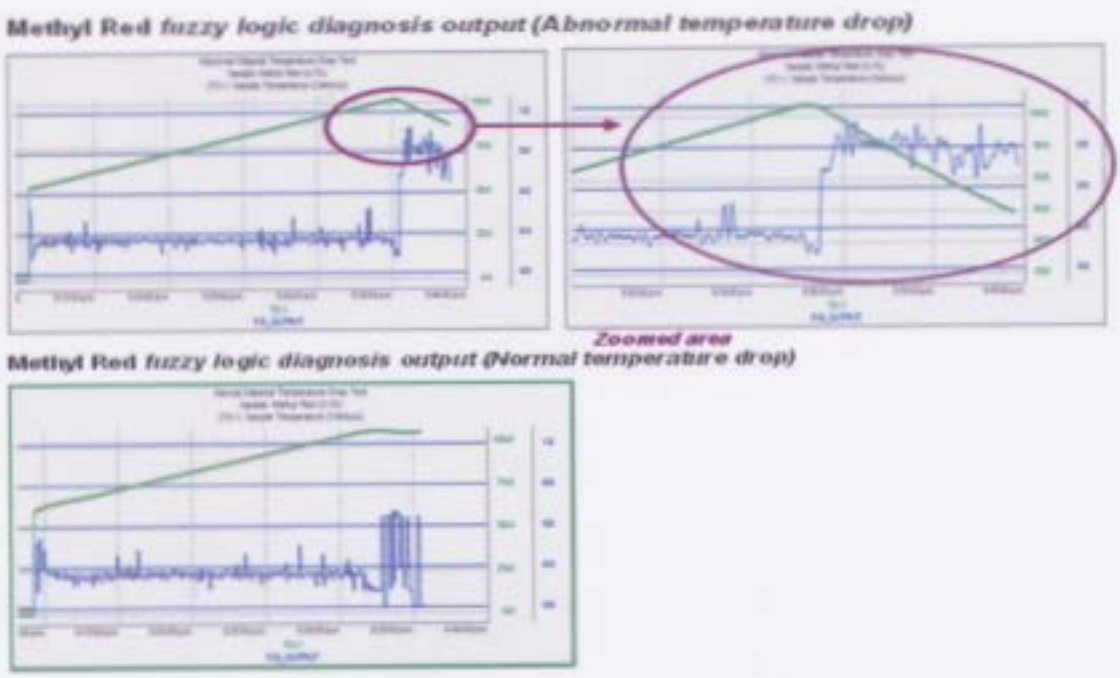


Figure 53 Sample 3: methyl red (C15H15N3O2)

Summary

This chapter demonstrates application of methodology and G2 based tool using two case studies. The first case study is carried out using simulated data of a micro steam power unit and the second case study is performed in a real process environment. Both case studies have presented acceptable outcomes. The capability and feasibility of the proposed methodology and applicability of the tool are also verified through case studies.

Chapter 6

RESULTS AND DISCUSSION

6.1 Fault Diagnosis Function

In chapter 5, two test cases are demonstrated to study the performance of fault diagnosis function, which is the essential part of the proposed SIS. The outcome of developed application is flexible, optimal and not dependent on the crisp input numbers. For example, in case study 1, a conventional alarm can also be added into the system to monitor boiler steam pressure whose limit is set at 625 kPa. As a consequence, both alarms (new alarm and alarm based on FIS) will be raised at some point when steam pressure reaches critical condition around 625 kPa. However, the fault event which causes this critical condition occurs in an uncertain environment. For example, when boiler steam pressure reaches 500 or 700 kPa, the conventional alarm will fail to provide a fast and proper notice to an operator in such environment. With the assistance of fuzzy logic inference system, the developed application will help to avoid this situation and maintain the capability of predicting the abnormal operating condition. The results are acceptable for current safety requirement.

Since this is a knowledge-based approach, a variety of information needs to be collected prior to testing. For example, what is the applicable range of ROC, what is the special pattern sample temperature as illustrated, or how long the abnormal situation lasts. This information is necessary for developing the application and performing the test. There is no related system models developed in the application.

Three industrial data case study results have been demonstrated from the previous section. Compared with normal process condition, there is a clear difference in the developed FIS responses for two (normal and abnormal) situations visually. This clarity highlights the importance that the proposed methodology and corresponding tool is able to detect undesired events.

The sampling rate for both case studies is 1 sample per 5 seconds. Since trend pattern defined for each fault event is composed of three primitives, the period between fault event happening and first detection response would be more than 15 seconds (5 seconds per primitive * 3 primitives). According to the outcomes from case study 2, the abnormal process condition (temperature drop) is first identified in around 30 seconds. If this is not acceptable, it can be improved by increasing sampling rate.

Noise hasn't been a serious issue in case study 2. As may be seen in Figure (51 - 53), the thermocouple TC-1 output is very smooth. This is because of the built-in filter of the GDA application, the physical isolation of TC-1 and good lab environment.

It should be noted that the FIS application for three samples studied in case study 2 varies. Input and/or output membership functions have been modified for each sample. This also means that the developed application can only be effective to the studied system. The change of material or environment would affect the accuracy of FIS outcome. Therefore, each system specific FIS needs to be defined.

6.2 Safety Instrumented System

The general fault diagnosis function can be implemented in a variety of safety systems. However according to the requirements of related system, it needs to be modified to satisfy the specified system goals. For example it might be used to predict and detect the overpressure hazards in offshore oil gas platform by revising FIS interface (input and output) and adjusting membership functions.

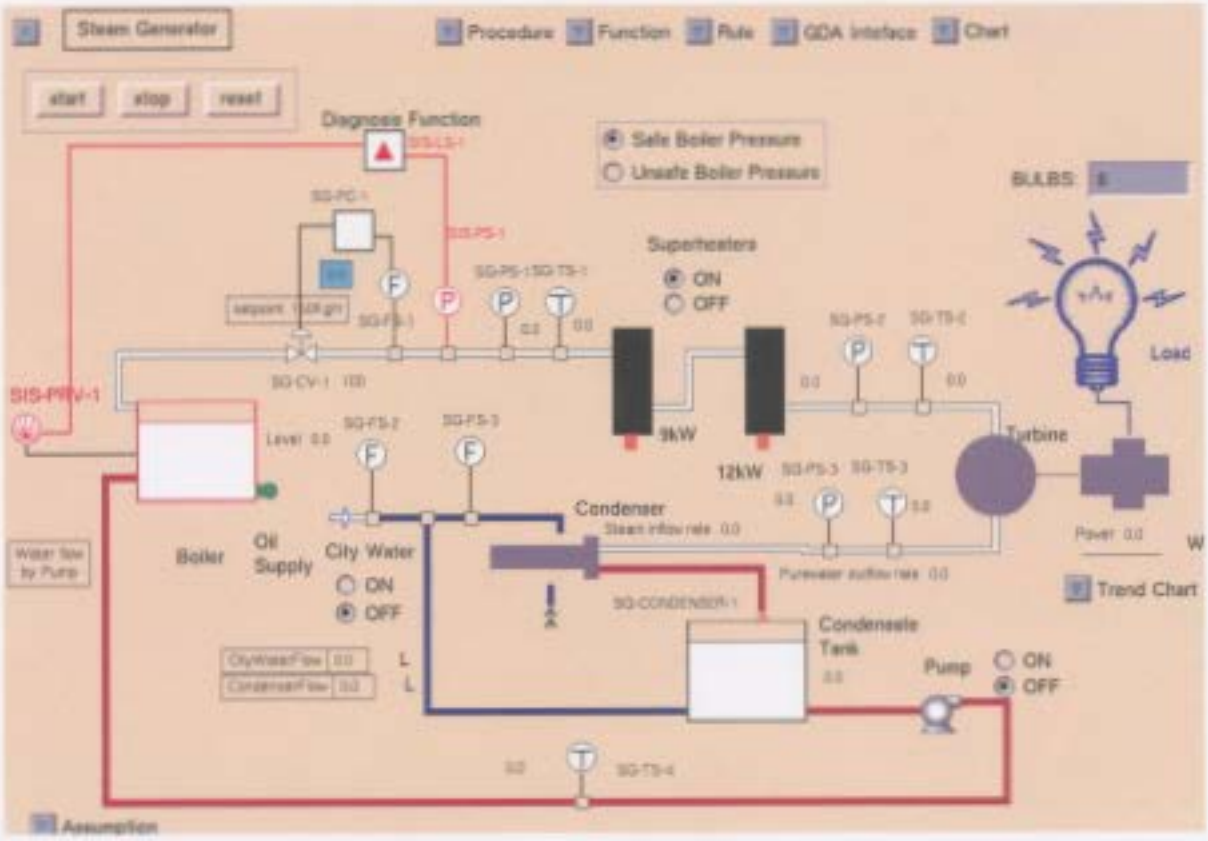


Figure 54 The snapshot of steam power unit simulator after adding SIS

Figure 54 illustrates the snapshot of steam power unit simulator, which is illustrated in chapter 3, after adding a SIS. This SIS is composed by a pressure sensor (SIS-PS-1), a logic solver (SIS-LS-1), and pressure relief valve (SIS-PSV-1). There are no process components shared by SIS and existing BPCS. The input of this SIS is the steam pressure out of the boiler obtained from SIS-PS-1. The logic solver is developed to perform the fault diagnosis function. The decision regarding the knowledge-based fault event is made by the outcome of FIS. The pressure relief valve (SIS-PSV-1) is a relief valve, which is used to control or limit the pressure in the steam boiler. After fault event decision is made, an alarm will be raised first in order to notify the operator. If it is not responded during any given period, the relief valve will start to release the pressurized steam out of the boiler. As a consequence, the steam pressure in the boiler will be decreased until the fault is noticed by the operator.

A similar SIS can also be added to the ARSST device. In second case study, fault diagnosis function is tested using a thermocouple, which is shared with BPCS. This can be accepted during current testing stage. However if this function is implemented in a SIS, an extra thermocouple is required in order to separate BPCS from SIS.

To reach the desired safety objective, safety analysis should be performed after applying SIS to existing process system using improved combining technique introduced in Chapter1. The performance of fault diagnosis function can be assessed through this analysis. A judgment will be made whether this general safety function is acceptable or improvement is recommended.

Summary

This chapter explains and discusses the results from previous case studies. The advantage and possible improvement of fault diagnosis function are discussed. It is the essential part of the developed safety instrumented system. This chapter also demonstrates how to implement general fault diagnosis function in a specific safety system.

Chapter 7

CONCLUSIONS AND FUTURE WORKS

7.1 Conclusions

Most process systems are inherently hazardous and often processing flammable and reactive materials at high temperature and pressure. Organizations have used a variety of techniques to control and reduce the risks posed by these hazards. Safety instrumented system (SIS) is a widely recognized tool by a number of industry sectors. In general, SIS aims to improve the process safety. Safety function (SF) can be considered as a method to define the functional relationship between inputs and outputs in SIS and is developed to implement SIS. Safety analysis is carried out to measure the likelihood of SIS performing the intended safety functions via predetermined safety integrity level (SIL). Fault diagnosis function is a common safety function, which works as an approach to develop a general SIS suited to various process systems. The primary focus of this thesis has been to design and implement a methodology of developing real-time SIS with general fault diagnosis function.

The proposed methodology in this thesis is divided into three stages: system modeling and simulation, knowledge-based fault diagnosis and G2 application development. The first stage provides information and platform for testing the performance of real-time general fault diagnosis function. It can help developers to escape the potential hazards of modifying real process system. The second stage is the essential part of the methodology and based on expert knowledge. This knowledge-bases feature, given its human-like-reasoning nature, is easy to understand and implement. This stage is comprised of two important components: process trend recognition and fuzzy logic system. The first component establishes the relationship between sensor trends and process operations. The process trend is identified from discrete real-time process data. The second component is able to map the expert knowledge with process trends using *if-then* rules. It also handles the uncertainty caused by first component.

The third stage is to develop a computer-aided application, implementing previous two stages, on the platform of G2 expert system platform using GDA (*G2 Diagnostic Assistant*) components.

The proposed methodology and G2-based application have proven to be of great advantage, demonstrated through two case studies. It provides a fundamentally simple way to handle complex process systems without making itself exceedingly complex. It is straightforward, flexible, and easy to develop and understand. Moreover, it is a critical part of protection systems in a variety of process operations.

This thesis covers all the details about design and implementation of real-time SIS with fault diagnosis function, including design methodology, computer application development and test case studies. It can be seen from this thesis that the studied SIS has the capability of capturing the knowledge-based fault events.

7.2 Future Works

The following recommendations have been suggested for the future improvement of the proposed works:

i. The accuracy of primitive identification can be improved. Since this is a real-time approach, the fast response for capturing the primitive from input sensor data is necessary. However this could lead to the possibility of highly unstable outcomes and thus has an effect on the accuracy of real-time process trend analysis. One possible solution to this limitation is to introduce a redundancy majority voting system: *Two-out-of-Three* (2oo3). For example, in order to identify current primitive, three *Fixed Window Discrete Data Primitive Identification* systems can be used at the same time. The final output will be decided by the majority vote.

ii. FIS method could be improved by defining more trend patterns. In both case studies, only one trend pattern is pre-defined. Due to the uncertainty of current real-time primitive identification system, some events might be hard to identify by only single trend pattern. A possible solution to this problem is to add extra trend pattern in order

to detect one event. Consequently this will increase the complexity of application development. However the knowledge-based fault event could be fully explained with more information. Furthermore, it will reduce the possibility of failing to identify desired fault.

iii. As knowledge-based fault diagnosis method is data-driven, the performance is dependent on the quality of expert knowledge and frequency of data processing. This could be further strengthened by integrating such a data-driven method with a simple model-based method in the future research, or adding additional sensors to acquire more knowledge about the system.

REFERENCES

Crowl, D. A., & Louvar, J. F. (1991). Chemical Process Safety – Fundamentals with Applications. *Chemical Engineering Science.46*(4).

Cusimano, J., & DiNapoli, L. (2001). Selecting Pressure Transmitters for Safety Instrumented Systems. *Journal of Chemical Engineering, 107*(6), 86-91.

Dash, S., Maurya, M. R., Venkatasubramanian, V., & Rengaswamy, Raghunathan. (2003). Fuzzy-logic based trend classification for fault diagnosis of chemical processes. *Journal of Computers and Chemical Engineering, 27*(3), 347-362.

Dash, S., Maurya, M. R., & Venkatasubramanian, V. (2004). A Novel Interval-Halving Framework For Automated Identification of Process Trends. *AIChE Journal, 50*(1), 149-162.

El-Shal, S. M., & Morris, A. S. (2000). A Fuzzy Expert System for Fault Detection in Statistical Process Control of Industrial Processes. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews, 30*(2), 281-289.

Foord, A. G., Gulland, W. G., Howard, C. R., Kellacher, T., & Smith, W. H. (2004). Applying the latest standard for functional safety – IEC 61511. *Institution of Chemical Engineers Symposium Series,* 150, 348-369.

Frankhauser, H. R. (2001). Safety Functions versus Control Functions. *SAFECOMP 2001*, 66-74.

Frank, P. L. (1986). Loss Prevention in the Process Industries. London: Butterworths, p. 343.

Goring, C.J. (1993). Safety first-applying computer-based safety systems in the offshore environment. *Computing & Control Engineering Journal, 4* (6), 245-252.

Gruhn, P. (1999). Safety Instrumented System Design: Lessons Learned. *Journal of Process Safety Progress, 18* (2), 156-160.

Harrell, C. (1998). Process Simulation vs. System Simulation. *IEEE Information Technology Conference*, 41-44.

Harris, C. J., Hong, X., & Gan, Q. (2002). Adaptive Modeling Estimation and Fusion from Data. New York: Springer.

Hendershot, D. C., & Dowell, A. M. (2002). Simplified Risk Analysis – Layer of Protection Analysis (LOPA). *Institution American Institute of Chemical Engineers 2002 National Meeting.*

Howell, J. (1994). Model-based Fault Detection in Information Poor Plants., *Automatica Journal , 30* (6), 929-943.

International Electrotechnical Commission 61508. (1998-2000). Functional Safety of Electrical/Electronic/Programmable Electronic Systems in Safety-related Applications, Parts 1-7.

International Electrotechnical Commission 61511. (1998-2003). Functional Safety – Safety Instrumented Systems for the Process Industry Sector, Parts 1-3.

Kirkwood, D., & Tibbs, B. (2005). Developments in SIL Determination. *IEEE Computing and Control Engineering*, 21-27.

Kosmowski, K. T. (2006). Functional safety concept for hazardous systems and new challenges. *Journal of Loss Prevention in the Process Industries*, 19, 298-305.

Lo, C. H., Wong, Y. K., & Rad, A. B. (2006). Intelligent System for Process Supervision and Fault Diagnosis in Dynamic Physical Systems. *IEEE Transactions on Industrial Electronics, 53*(2), 581-592.

MacDonald, D. (2003). Practical Industrial Safety, Risk Assessment and Shutdown Systems for Industry. New York: Newnes.

Marcellus, R. L. (1997). Evaluation of a nonstationary policy for statistical process control. *Proceedings of the 1997 6th Annual Industrial Engineering Research Conference*, 89-94.

Marszal, Edward M., & Mitchell, Kevin J. (2003). Defining Safety Instrumented Functions. *Safety Instrumented Systems for the Process Industry Conference*, 1-16.

McMath, I., & Kingman, B. (2005). Integrated, Scalable Safety Instrumented System. *Engineering Technology Journal, 8* (5), 47.

Mervyn, R. C. (2004). Safety Instrumented Systems: Reconciling the Traditional Prescriptive Approach in APIRP 14C with More Recent Risk-Based Approach Methods in IEC 61511. *2004 SPE Annual Technical Conference and Exhibition*, 237-240.

Mohamed, L., & Ibrahim, A. S. (2002). Model-Based Fault Diagnosis via Parameter Estimation Using Knowledge Based and Fuzzy logic Approach. presented at IEEE MELECON 2002, Cairo, Egypt.

Monsef, H., Ranjbar, A. M., & Jadid, S. (1997). Fuzzy Rule-based Expert System for Power System Fault Diagnosis. *IEE Proceedings: Generation, Transmission and Distribution, 144* (2), 186-192.

Mouss, D., Mouss, H., Khedri, K., & Mouss, N. (2004). Simulation of an industrial application by the expert systems generator G2. *5th International Conference on Quality, Reliability, and Maintenance*, 153-156.

Ogata ,K. (2004). Modern Control Engineering. Upper Saddle River, N.J. : Prentice Hall.

Rothschild, M. (2004). Fault Tree and Layer of Protection Hybrid Risk Analysis. *Journal of Process Safety Process ,.23*(3), 185-190.

Sharif, M. A., & Grosvenor, R. I. (1998). Process plant condition monitoring and fault diagnosis. *Journal of Process Mechanical Engineering, 212* (E1), 13-30.

Siegel, D., Abidi, K., & Anderson, R. (2004). Abnormal Condition Management. published by Gensym Corporation.

Summers, A. E. (2006). IEC 61511 and the capital project process – A protective management system approach. *Journal of Hazardous Materials, 130*(1-2), 28-32.

UK Health and Safety Executive. (1987). Programmable Electronic Systems in Safety Related Applications, Part 1 – An Introductory Guide, Part 2 – General Technical Guidelines. England

Wiegerinck, Jan A. M. (2002). Introduction to the Risk based design of Safety Instrumented Systems for the process industry. *Seventh International Conference on Control, Automation, Robotics And Vision (ICARCV'02)*, 1383-1391.

Wolfram, A., Füssel, D., Brune, T., & Isermann, R. (2001). Component-based Multi-model Approach for Fault Detection and Diagnosis of a Centrifugal Pump. *Proceedings of the American Control Conference, 6*, 4443-4448.

APPENDIX A

Source Code of G2 Structured Procedural Language

Boiler Simulation Core Code

```
**********************************************************************
 boiler-sim()
{**********Definition**************}
FIN1, FOUT1: float;
in-flow-rate: float = 200.0;
sim-period : integer = 5; {update every 5 seconds}
blr-temp: quantity = 159.1;
before-turbine-temp-heater-off: quantity=159;
before-turbine-temp: quantity;
after-turbine-temp-heater-off: quantity = 106.7;
after-turnbine-temp: quantity ;
level-boiler: quantity = the level of sg-blr-1 ;
minimum-boiler-level: quantity =0.1;
kgh-to-m3s: quantity = 3600000;
unsafe-blr-pres: quantity;
blr-pres-release-sp1: quantity = the pressure-sp of sg-pr-1 ;
blr-pres: quantity;
{**********Definition End********************}
begin
  conclude that simulate-time = simulate-time + sim-period;
  conclude that state-update-simulate-time = state-update-simulate-time + sim-period;
  if the boiler-enable of sg-blr-1 is false then return;
     if safe-boiler-pres then begin
     collect data
      {The reason to add/250/is that I don't want simulation value start from zero}
     FIN1 = simulate-blr-pres-calculate (simulate-time + 250);
     FOUT1 = blr-flow-calculate (simulate-time + 250);
    end;
 end
  else begin
     collect data
     FIN1 = unnormal-sim-pres;
     FOUT1 = 65+0.152*unnormal-sim-pres;
     end;
   end;
{----------flow rate set up----------}
conclude that the steam-outflow-rate of sg-blr-1 = FOUT1;
{call  sg-pc1-calculate (FOUT1 );}
```

```
{----------flow rate set up end--------}

{--------valve lift position calculate----------}
conclude that sg-lift-position = valve-position-calculate (simulate-time + 250 );
if sg-lift-position >= 99.4 then conclude that sg-lift-position =99.4 else if sg-lift-position <10 then conclude that sg-lift-position =10;
conclude that the position of sg-cv-1 = sg-lift-position;
{----------valve lift position calculate end----------}
{------------First high pressure release--------}
{if blr-release-on  then conclude that the boiler-pressure of sg-blr-1= the boiler-pressure of sg-blr-1 - 5 else conclude that the boiler-pressure of sg-blr-1 = FIN1; old backup}
if blr-release-on  then conclude that the boiler-pressure of sg-blr-1= the boiler-pressure of sg-blr-1 - 5;
{---------add these code for unsafe pressure simulation----------}
if not(safe-boiler-pres) then begin
conclude that the boiler-pressure of sg-blr-1 = unsafe-pressure-sim (unsafe-blr-pres-sim , UNSAFE-BLR-SIM-TIME );
conclude that UNSAFE-BLR-SIM-TIME = UNSAFE-BLR-SIM-TIME + sim-period;
{--------------add for the test----------------}
 if (UNSAFE-BLR-SIM-TIME >= 90) then conclude that safe-boiler-pres is true;
{--------------add for the test  end----------------}
end
else begin
conclude that the boiler-pressure of sg-blr-1 = FIN1;
conclude that  unsafe-blr-pres-sim= FIN1 +105.0;
end;
{--------- unsafe pressure simulation end----------}
if the boiler-pressure of sg-blr-1 <= 600 and blr-release-on then begin
conclude that blr-release-on = false;
inform the operator that "Boiler Pressure has reached to a safety point , pressure gauge will close";
change the background icon-color of sg-pr-1 to white;
end;
if the boiler-pressure of sg-blr-1 >= blr-pres-release-sp1 and not (blr-release-on) then begin
conclude that blr-release-on = true;
inform the operator that "Boiler Pressure has reached to a critical point , pressure gauge will start to release pressure";
change the background icon-color of sg-pr-1 to red;
end;
conclude that the p-output of sg-ps-1= the boiler-pressure of sg-blr-1 + random(-.025,.025) * the boiler-pressure of sg-blr-1 ;
{------------First high pressure release End--------}
{----------------pressure after super heater----------}
conclude that the p-output of sg-ps-2= pres-after-superheater-on (simulate-time + 250 );
{----------------pressure after turbine boiler----------}
conclude that the p-output of sg-ps-3= 197;
{----------------Power calcuate----------}
{won't use the one from thermalsynamics formula}
{call turbine-power-calculate (condenser, the t-output of sg-ts-2, the t-output of sg-ts-3, the p-output of sg-ps-3);}
```

```
if not (state-update-simulate-on) then begin
conclude that the power of sg-tubine-1 = turbine-power-estimate (FIN1, simulate-time + 250 );
conclude that normal-power =0;
end
else
begin
 conclude that the power of sg-tubine-1 = turbine-noise-estimate (FIN1, state-update-simulate-time);
 conclude that normal-power = turbine-power-estimate (FIN1, simulate-time + 250 );
end;
{calculate condenser level}
call condenser-sim (condenser, simulate-time + 250 );
{----------level of boiler calculation----------}
{conclude that the level of sg-blr-1 =level-boiler+((in-flow-rate* the pump-open-state of sg-pump-1 - the steam-outflow-rate of sg-blr-1*0.8 )/ kgh-to-m3s )* sim-period / the area of sg-blr-1;}
{-------use a first order equation to calculate the boiler level---}
conclude that the level of sg-blr-1 = level-boiler + blr-level-estimate ((in-flow-rate* the pump-open-state of sg-pump-1 - the steam-outflow-rate of sg-blr-1*0.8 )/ kgh-to-m3s, simulate-time + 250 );
if the level of sg-blr-1 > minimum-boiler-level then conclude that the water-inflow-enable of sg-blr-1 is false ;
if the level of sg-blr-1 < minimum-boiler-level and the water-inflow-enable of sg-blr-1 = false then begin
 conclude that the water-inflow-enable of sg-blr-1 is true;
 inform the operator that "The level of Boiler is less than [minimum-boiler-level]m , Please turn on pump";
end;
{----------Temperature set up----------}
conclude that the steam-temperature of sg-blr-1 = blr-temp-calculate (FIN1);
conclude that the t-output of sg-ts-1 = the steam-temperature of sg-blr-1 + random(-0.05,0.05) * the steam-temperature of sg-blr-1 ;
if super-heater-on then begin
before-turbine-temp = temp-after-superheater-on (the steam-temperature of sg-blr-1, simulate-time + 250 );
conclude that the t-output of sg-ts-2 = before-turbine-temp + random(-0.05,0.05) * before-turbine-temp ;
after-turnbine-temp = temp-after-turbine-superheater-on (the steam-temperature of sg-blr-1, simulate-time + 250 );
conclude that the t-output of sg-ts-3 = after-turnbine-temp + random(-0.05,0.05) * after-turnbine-temp ;
end
else begin
conclude that the t-output of sg-ts-2 = before-turbine-temp-heater-off + random (-0.9,0.9);
conclude that the t-output of sg-ts-3 = after-turbine-temp-heater-off +random(-0.1,0.1);
end;
conclude that the t-output of sg-ts-4 = 60.8 + random(-7,7);
{----------Tubine Power Calculation----------}
conclude that the volts of sg-tubine-1 = random (67.1,79.7);
conclude that the amps of sg-tubine-1 = random (10.7,11.6);
{----------City water input (in Liter ) set up----------}
if city-water-on then
begin
conclude that city-water-total-flow = city-water-total-flow +130;
conclude that the source-total-flow of city-water = city-water-total-flow;
```

```
end;
 if process-sim-active then start boiler-sim () after sim-period;
end
```

****

Start Simulation Code

****

```
start-sim (Button: class uil-button, W : class item, Itm: class item)
begin
if process-sim-active then
   return;
call reset-sim(FALSE);
call gdl-enable-data-input (W);
conclude that process-sim-active = TRUE;
{--------Add for process trend test----------}
start process-trend-trial ();
start boiler-sim();
{wait for 5 seconds to start codensor sim}
{start condemsate-tank-sim...();}
end
```

****

Reset Simulation Code

****

```
reset-sim(do-block-reset: truth-value)
begin
{----------Add for process trend trial ----------}
conclude that process-trend-time = 0;
conclude that PROCESS-SIM-ACTIVE is false;
{abort pid-evaluator;}
{change the text of the manual-position of cv-1 to "none";}
{conclude that the position of cv-1 = 10.0;}
conclude that the level of sg-blr-1 = 0.3;
```

```
conclude that the boiler-enable of sg-blr-1 = true;
conclude that the water-inflow-enable of sg-blr-1 = false;
conclude that the sp of sg-pc-1 = 160;
conclude that blr-release-on =false;
conclude that UNSAFE-BLR-SIM-TIME =0;
conclude that the level of sg-cond-tank =0.1;
conclude that the area of sg-cond-tank =0.359;
change the background icon-color of sg-pr-1 to white;
conclude that SIMULATE-TIME =0;
{simulation pressure parameter set up}
conclude that  wn= wn-calculate ();
conclude that OMEGA= omega-calculate ();
{simulation flow parameter set up}
conclude that  wn-flow= wn-flow-calculate ();
conclude that OMEGA-flow= omega-flow-calculate ();
conclude that state-update-simulate-on = false;
conclude that blr-pressure-sp =640;
conclude that the message-contents of number-of-bulbs = "10";
{conclude that  the position of mv-1 =100;}
{conclude that the sp of lc-1 =5;}
{conclude that the threshold of ob-1 =1 + the sp of lc-1;}
{conclude that level-estimate = 10.0;}
{conclude that the error of lc-1 = 0.0;}
{conclude that the error-1 of lc-1 = 0.0;}
{conclude that the valve-constant of cv-1 = 3.0;}
{conclude that flow-sensor-bias = 0.0;}
if do-block-reset then
   call gdl-reset-all-gdl-objects (gfr-default-window, false,False);
end
**********
```

Reset Simulation Call Code

****

```
reset-sim-call(itm1: class item, itm2: class item, itm3: class item)
begin
call reset-sim(true);
end
**********
```

Stop Simulation Call Code

```
**********
stop-sim(itm1: class item, itm2: class item, itm3: class item)
begin
conclude that PROCESS-SIM-ACTIVE = FALSE;
call gdl-disable-data-input(gfr-default-window);
end
**********
```

Process Power Load Code

```
**********
process-power (G: class item, W: class ui-client-item, D-or-W: item-or-value, B: item-or-value, action-queue: item-
or-value) = (text)
power-value: quantity;
begin
{-- The validation method for an object validates the contents of the object, according to the format specification
referred to by the object's uil-format-specification attribute.
G is the UIL object on which to run the validation method
W is the window on which D-or-W is managed
D-or-W is the dialog or workspace on which the object is managed
B is the button that initiated the validation action for the object (optional)
action-queue is the list of pending actions for the dialog initiated by B (optional)
A text string is returned, which is "OK" if the validation succeeded, and a built-in error string otherwise. --}
power-value = call utl-convert-from-text-to-quantity(the text of G);
if power-value > 10 or power-value < 0 then inform the operator that "The maximum number of bulbs is 10" else
begin
inform the operator that "The number of bulbs has been modified to [power-value]";
```

```
if power-value = 8 then conclude that blr-pressure-sp = 629 else if power-value = 7 then conclude that blr-pressure-sp = 614;
conclude that defined-power = power-value;
end;
return "OK";
end
**********
```

Procedure if Primitive A is first to get

```
**********
primA-1 (actionBlock: class gdl-generic-action)
begin
wait for 1 seconds;
conclude that s1=0;
conclude that s2=0;
conclude that s3=0;
conclude that s1= trend-primit-matrix ("A","B");
if s1 > 0 then begin
conclude that p1=false;
conclude that r=1;
conclude that p2= true;
end
else begin
conclude that si=0;
end;
end
**********
```

Procedure if Primitive A is second to get

```
**********
primA-2 (actionBlock: class gdl-generic-action)
begin
wait for 1 seconds;
conclude that s2= trend-primit-matrix ("A","B");
```

```
if s2 > 0 then begin
conclude that s=s+s2;
conclude that p2=false;
conclude that r=2;
conclude that p3= true;
end
else
begin
conclude that p2= false;
conclude that p1 = true;
conclude that si=0;
end;
end
**********
```

Procedure if Primitive A is third to get

```
**********
primA-3 (actionBlock: class gdl-generic-action)
begin
wait for 1 seconds;
conclude that s3= trend-primit-matrix ("A","G");
if s3 > 0 then begin
conclude that r=3;
conclude that s=s+s1;
conclude that si=(s1+s2+s3)/r;
conclude that p3=false;
conclude that p1= true;
end
else
begin
conclude that si=(s1+s2)/ 3;
conclude that p3= false;
conclude that p1 = true;
```

```
end;
end
**********
```

Procedure to calculate trend based on the acquired primitive value

```
**********
get-trend(reset-trend: truth-value)
begin
if trend-primit = "A" then begin
conclude that trend-a = true;
change the icon-color of prima-object to red;
wait for 1 second;
conclude that trend-a = not (trend-a);
change the icon-color of prima-object to blue ;
end;
if trend-primit = "B" then begin
conclude that trend-b = true;
change the icon-color of primb-object to red;
wait for 1 second;
conclude that trend-b = not (trend-b);
change the icon-color of primb-object to blue ;
end;
if trend-primit = "C" then begin
conclude that trend-c = true;
change the icon-color of primc-object to red;
wait for 1 second;
conclude that trend-c = not (trend-c);
change the icon-color of primc-object to blue ;
end;
if trend-primit = "D" then begin
conclude that trend-d = true;
change the icon-color of primd-object to red;
wait for 1 second;
```

```
conclude that trend-d = not (trend-d);
change the icon-color of primd-object to blue ;
end;
if trend-primit = "E" then begin
conclude that trend-e = true;
change the icon-color of prime-object to red;
wait for 1 second;
conclude that trend-e = not (trend-e);
change the icon-color of prime-object to blue ;
end;
if trend-primit = "F" then begin
conclude that trend-f = true;
change the icon-color of primf-object to red;
wait for 1 second;
conclude that trend-f = not (trend-f);
change the icon-color of primf-object to blue ;
end;
if trend-primit = "G" then begin
conclude that trend-g = true;
change the icon-color of primg-object to red;
wait for 1 second;
conclude that trend-g = not (trend-g);
change the icon-color of primg-object to blue ;
end;
end
```

Procedure to acquire the output from external sensor

```
**********************************************************************
process-sensor-input(new-sensor-input: class sensor-input )
begin
 conclude that sensor-reading1 =the sensor-point1 of new-sensor-input;
 conclude that sensor-reading2 =the sensor-point2 of new-sensor-input
```

end

****

Method to simulate the condenser tank

****

condenser-sim (itm1: class item, simulation-time: quantity)

blr-flowrate: quantity = the steam-outflow-rate of sg-blr-1;

steam-inflow: quantity;

sim-period: integer = 5;

kgh-to-m3s: quantity = 3600000;

level-condense: quantity = the level of sg-cond-tank;

water-outflow-rate: quantity = 200.0;{accordiing to the parameter I have defined in boiler}

begin

steam-inflow = (blr-flowrate - random (10,15)) * (1 - the steam-exhausted-rate of sg-tubine-1);

if steam-inflow < 0 then steam-inflow=0;

conclude that the steam-inflow-rate of sg-condenser-1 =steam-inflow;

conclude that the purewater-outflow-rate of sg-condenser-1 = condenser-output-estimate (steam-inflow, simulation-time);

conclude that the level of sg-cond-tank = level-condense + cond-level-estimate (((the purewater-outflow-rate of sg-condenser-1 - water-outflow-rate * the pump-open-state of sg-pump-1)/ kgh-to-m3s) , simulation-time)

end

Procedure to start the ARSST monitor

****

start-arsst-sim (Button: class uil-button, W : class item, Itm: class item)

begin

if arsst-sim-active then

  return;

call reset-arsst-sim(FALSE);

call gdl-enable-data-input (W);

conclude that arsst-sim-active = TRUE;

conclude that is-heater-on = true;

```
start arsst-sim();
{start condemsate-tank-sim...();}
End

**********
```

Procedure to stop the ARSST monitor

```
**********
stop-arsst-sim(itm1: class item, itm2: class item, itm3: class item)
begin
conclude that ARSST-SIM-ACTIVE = FALSE;
call gdl-disable-data-input(gfr-default-window);
end
**********
```

Procedure to reset the ARSST monitor

```
**********
reset-arsst-sim(do-block-reset: truth-value)
begin
conclude that ARSST-SIM-ACTIVE is false;
if do-block-reset then
   call gdl-reset-all-gdl-objects (gfr-default-window, false,False);
end
**********
```

APPENDIX B

Source Code of C Language in GSI

```
/*****************************************************************************
: alarmer.c -- Sample GSI Bridge Code for use with gsi_exam.kb
: When enabled, sends a sample alarm (as if from some external system)
: every so often to G2 for processing.
: This file contains the standard GSI toolkit functions required for
: any GSI application and several other functions coded to support
: part of the sample knowledge base `gsi_exam.kb' provided with G2.
: This file requires the standard GSI libraries and the sample main
: module 'gsi_main.c' to create the executable `alarmer'.
: This file conforms to GSI 4.0.
: Created 10mar95 by paf!
: Modified by kelvin . I have made this code a example code to transfer datapoint between
: GSI and G2
*****************************************************************************/
//#define GSI_USE_WIDE_STRING_API
#include <stdio.h>
// Program needs this DLL to communicate with GSI
#define GSI_USE_DLL
#include "gsi_main.h"
#include "cbw.h"
#define TCPIP_PORT_NUMBER  22041
#define G2_ALARM_ATTR_COUNT 5
#define MESSAGE_SYMBOL     "MESSAGE"
#define PRIORITY_SYMBOL    "PRIORITY"
#define DATA_POINT_SYMBOL  "DATA-POINT"
#define SENSOR_POINT1_SYMBOL "SENSOR-POINT1"
```

```
#define SENSOR_POINT2_SYMBOL  "SENSOR-POINT2"
#define FAILURE  1
#define WARNING  2
#define INFORM   3
#define FAILURE_SYMBOL "FAILURE"
#define WARNING_SYMBOL "WARNING"
#define INFORM_SYMBOL  "INFORM"
typedef struct {
 long   data_point_tag;
 long   priority;
 char  *message;
 float sensor_point1_tag;
 float sensor_point2_tag;
} p3_alarm;
static function_handle_type  process_sensor_input;
static int       sensor_enabled = FALSE;
static p3_alarm  sample_sensor = { 99,
 WARNING,
 "test string",
 23.2 ,
 11.2};
static gsi_item *g2_alarm_ptr;
static gsi_item  g2_alarm;
extern declare_gsi_rpc_local_fn(enable_sensor_input);
extern declare_gsi_rpc_local_fn(disable_sensor_input);
const int BoardNumber=1;
const int channel=3;
const int channel0=0;
main(argc, argv)
   int   argc;
   char *argv[];
```

```
{
   // test data card can be passed or not
 char chipName[25];
   if ( cbGetBoardName(BoardNumber, chipName) == 0)
{
     printf("Data Card Test Pass \n");
}
 gsi_initialize_for_win32(NULL, NULL);
 gsi_set_include_file_version(GSI_INCLUDE_MAJ_VER_NUM,GSI_INCLUDE_MIN_VER_NUM,
GSI_INCLUDE_REV_VER_NUM);
GSI_SET_OPTIONS_FROM_COMPILE();
gsi_set_option(GSI_TRACE_RUN_LOOP);
//gsi_set_option(GSI_PROTECT_INNER_CALLS);
/*
 * Initialize GSI and enter the event handler loop.
 */
gsi_start(argc, argv);
} /* end main */
/*****************************************************************************
: RPC FUNCTION: enable_alarming
: This function, which is declared as RPC-invocable from a G2 process, sets a
: flag that enables the transmission of alarm objects to G2.
: RPC Arguments (0)
: Note, this function does not support being invoked via a 'call', and must
: be invoked with a 'start' action.
*****************************************************************************/
void enable_sensor_input(arg_array, count, call_index)
 gsi_item *arg_array;
 gsi_int   count;
 gsi_int   call_index;
{
```

```
 sensor_enabled = TRUE;
} /* end enable_alarming */
/*****************************************************************************
: RPC FUNCTION: disable_alarming
: This function, which is declared as RPC-invocable from a G2 process, sets a
: flag that disables the transmission of alarm objects to G2.
: RPC Arguments (0)
: Note, this function does not support being invoked via a 'call', and must
: be invoked with a 'start' action.
*****************************************************************************/
void disable_sensor_input(arg_array, count, call_index)
 gsi_item *arg_array;
 gsi_int   count;
 gsi_int   call_index;
{
 sensor_enabled = FALSE;

} /* end disable_alarming */
/*****************************************************************************
: FUNCTION: send_alarm_to_g2
: This function sends an alarm to G2, based on an alarm structure defined by
: some hypothetical third party system.  Most of the work is in copying the
: contents of the alarm into a GSI structure for transmission to G2.
: Arguments (1):
:   alarm_ptr    A pointer to an alarm.
: Note, this could be made more efficient by using additional globals to
: maintain references to the attribute structures instead of using the API
: function attr_by_name for each every time this function is called.
*****************************************************************************/
void send_alarm_to_g2(alarm_ptr)
 p3_alarm *alarm_ptr;
```

```
{
/*
 * Copy the data from the p3_alarm into the g2_alarm.
 */
  // this is a test code , the thermalcouple reading will be sent to the G2
   unsigned short dataValue_tc1;
 float volts_tc1;
 unsigned short dataValue_pres;
 float volts_pres;
 set_int(attr_by_name(g2_alarm,DATA_POINT_SYMBOL),alarm_ptr->data_point_tag);
  switch(alarm_ptr->priority) {
    case FAILURE:
      set_sym(attr_by_name(g2_alarm,PRIORITY_SYMBOL),FAILURE_SYMBOL); break;
    case WARNING:
      set_sym(attr_by_name(g2_alarm,PRIORITY_SYMBOL),WARNING_SYMBOL); break;
    case INFORM:
      set_sym(attr_by_name(g2_alarm,PRIORITY_SYMBOL),FAILURE_SYMBOL); break; }
  set_str(attr_by_name(g2_alarm,MESSAGE_SYMBOL),alarm_ptr->message);
  // read thermal couple
    cbAIn(BoardNumber,channel,1,&dataValue_tc1);
cbToEngUnits(BoardNumber,1,dataValue_tc1,&volts_tc1);
  set_flt(attr_by_name(g2_alarm,SENSOR_POINT1_SYMBOL),volts_tc1);
  // read pressure
  cbAIn(BoardNumber,channel0,1,&dataValue_pres);
cbToEngUnits(BoardNumber,1,dataValue_pres,&volts_pres);
  set_flt(attr_by_name(g2_alarm,SENSOR_POINT2_SYMBOL),volts_pres);
/*
 * Send the alarm to G2 via RPC.
 */
  gsi_rpc_start(process_sensor_input,g2_alarm_ptr,current_context);
}
```

```
/*****************************************************************************
* The remaining functions are the standard GSI Toolkit functions required of all
* GSI applications. Those which are used preceed those which are stubbed out.
*****************************************************************************/
void gsi_set_up ()
{
 gsi_attr *attrs;
/*
 * Construct the alarm item to be used each time an alarm is
 * send to G2. No initial values are given for the attributes.
 */
 g2_alarm_ptr = gsi_make_items(1);
 g2_alarm = *g2_alarm_ptr;
 set_class_name(g2_alarm,"SENSOR-INPUT");
 attrs = gsi_make_attrs_with_items(G2_ALARM_ATTR_COUNT);
 set_attr_name(attrs[0],DATA_POINT_SYMBOL);
 set_attr_name(attrs[1],PRIORITY_SYMBOL);
 set_attr_name(attrs[2],MESSAGE_SYMBOL);
 set_attr_name(attrs[3],SENSOR_POINT1_SYMBOL);
 set_attr_name(attrs[4],SENSOR_POINT2_SYMBOL);
 set_attrs(g2_alarm,attrs,G2_ALARM_ATTR_COUNT);
/*
 * Declare local functions to be remotely invocable.
 */
 gsi_rpc_declare_local(enable_sensor_input,"ENABLE-SENSOR-INPUT");
 gsi_rpc_declare_local(disable_sensor_input,"DISABLE-SENSOR-INPUT");
}
gsi_int gsi_get_tcp_port()
{
 return(TCPIP_PORT_NUMBER);
}
```

```
gsi_int gsi_initialize_context (remote_process_init_string, length)
  char   *remote_process_init_string;
  gsi_int length;
{
/*
 *  Declare G2 procedure to be invocable from GSI.
 */
 printf("\n FInd TCP");
  gsi_rpc_declare_remote(&process_sensor_input,"PROCESS-SENSOR-
INPUT",NULL_PTR,1,0,current_context);
   return (GSI_ACCEPT);
}
void gsi_g2_poll()
{
 if (sensor_enabled)
    send_alarm_to_g2(&sample_sensor);
}
void gsi_shutdown_context()
{
 sensor_enabled = FALSE;
}
```

APPENDIX C

G2 Function Blocks

G2 Function Block	Explanation
omega-calculate()= -1 * ln (the pressure-startup-os of sg-blr-1) /sqrt(expt(3.1415926,2)+expt(ln(the pressure-startup-os of sg-blr-1),2))	This function is responsible to calculate the damping coefficient of second order equation of boiler steam pressure
wn-calculate()=3.1415926/(the pressure-startup-tp of sg-blr-1 * sqrt(1 - expt(omega,2)))	This function is responsible to calculate the undamped natural frequency of second order equation of boiler steam pressure
simulate-blr-pres-calculate(t)=(1 - sin(wn * sqrt(1 - expt(omega,2)) * t + arctan (sqrt(1 - expt(omega,2))/ omega)) * exp(-1 * omega * wn * t)/sqrt(1 - expt(omega,2)))* BLR-PRESSURE-SP + pres-white-noise	This function is responsible to calculate the simulated boiler steam pressure
omega-flow-calculate()= -1 * ln (0.12) /sqrt(expt(3.1415926,2)+expt(ln(0.12),2))	This function is responsible to calculate the damping coefficient of second order equation of boiler steam flow rate
wn-flow-calculate()=3.1415926/(the flow-startup-tp of sg-blr-1 * sqrt(1 - expt(omega-flow ,2)))	This function is responsible to calculate the undamped natural frequency of second order equation of boiler steam flow rate

```
simulate-blr-flow-calculate(t)=(1 - sin(wn-flow
    * sqrt(1 - expt(omega-flow,2)) * t + arctan
    (sqrt(1 - expt(omega-flow,2))/ omega-
    flow)) * exp(-1 * omega-flow * wn-flow *
    t)/sqrt( 1 - expt(omega-flow,2)))*the sp of
    sg-pc-1
```

This function is responsible to calculate the simulated boiler steam flow rate

```
turbine-power-estimate(p,t) = (126.1 - 1.281
    * p + 0.003435 * p * p) * (1 - exp(-0.025 *
    t))
```

This function is responsible to calculate the estimated turbine power

```
pres-after-superheater-on (t) = (simulate-blr-
    pres-calculate (t) *0.9942 - 19.85) * (1 -
    exp(-0.05 * t))
```

This function is responsible to calculate the simulated steam pressure after super heaters when super heaters are turned on

```
blr-temp-calculate(p) = 106.7+ 0.0038 * p +
    0.00027 * p * p
```

This function is responsible to calculate the simulated boiler steam temperature

```
temp-after-superheater-on(temp, t) = temp *
    1.0058 * (1 - exp(-0.05 * t))
```

This function is responsible to calculate the simulated steam temperature after super heaters when super heaters are turned on

```
valve-position-calculate(t) = 61.8+(59/ blr-
    pressure-sp
    )*exp(-0.0006*(t+250))*simulate-blr-pres-calculate
    (t) * (1 - exp(-0.2 * t))
```

This function is responsible to estimate the valve (SG-CV-1) position

```
unsafe-pressure-sim(p,t) =(p - 105*
    exp(-0.1*t))
```

This function is responsible to generate the unsafe steam pressure event

```
blr-level-estimate (q,t) = (1 / the area of sg-
    blr-1) * q * (1 - exp(-0.1 * t))
```

This function is responsible to estimate the water level in boiler

```
cond-level-estimate (q,t) = (1 / the area of
    sg-cond-tank ) * q * (1 - exp(-0.0127 * t))
```

This function is responsible to estimate the water level in condenser tank

```
pipe-delay-recalculate(pipeID, t) =(
if pipeID = 1
  then (1 - exp(-0.1 * t))
else if pipeID=2
  then (1 - exp(-0.0667 * t))
else if pipeID = 3
  then (1 - exp(-0.0571* t))
else
    (1 - exp(-0.0533* t))
)
```

This function is responsible to calculate the pipe delay according to the pipe ID

```
condenser-output-estimate(inflow,t) =
    random (0.85,0.92) * inflow * (1 -
    exp(-0.01484 * t))
```

This function is responsible to estimate the water flow rate out of condenser

```
temp-after-turbine-superheater-on (temp,t) =
    temp * random (0.673,0.686) * (1 -
    exp(-0.0025 * t))
```

This function is responsible to calculate simulated steam temperature after turbine when super heaters are turned on

```
Trend-Primit-Calculate(d, dd) =(
if (d >= 0.1 and dd <= - 0.1 )
  then ("D")
else if (d >= 0.1 and dd >= 0.1 )
  then ("B")
else if (d >= 0.1 and abs(dd) < 0.1 )
  then ("C")
else if (d <= - 0.1 and dd <= - 0.1 )
  then ("G")
else if (d <= - 0.1 and dd >= 0.1 )
  then ("E")
else if (d <= - 0.1 and abs(dd) < 0.1 )
  then ("F")
else
    ("A")
)
```

This function is responsible to calculate current primitive according to FDD and SDD

```
Trend-Primit-Matrix(prim1,prim2) =(
if (prim1 = prim2)
  then (1)
else if (prim1 = "A" and prim2 = "C")
  then (0.25)
else if (prim1 = "A"and prim2 = "F")
  then (0.25)
else if (prim1 = "B"and prim2 = "C")
  then (0.75)
else if (prim1 = "B"and prim2 = "D")
  then (0.5)
else if (prim1 = "C"and prim2 = "A")
  then (0.25)
else if (prim1 = "C"and prim2 = "B")
  then (0.75)
else if (prim1 = "C"and prim2 = "D")
  then (0.75)
else if (prim1 = "D"and prim2 = "B")
  then (0.5)
else if (prim1 = "D"and prim2 = "C")
  then (0.75)
else if (prim1 = "E"and prim2 = "F")
  then (0.75)
else if (prim1 = "E"and prim2 = "G")
  then (0.5)
else if (prim1 = "F"and prim2 = "A")
  then (0.25)
else if (prim1 = "F"and prim2 = "E")
  then (0.75)
else if (prim1 = "F"and prim2 = "G")
  then (0.75)
else if (prim1 = "G"and prim2 = "E")
  then (0.5)
else if (prim1 = "G"and prim2 = "F")
  then (0.75)
else
    (0)
```

This function is responsible to calculate similarity between two primitives according to the similarity table

APPENDIX D

G2 Rule Blocks

```
whenever the boiler-pressure P of sg-blr-1
    receives a value then conclude that the
    p-output of sg-ps-1 = P +
    random(-.025,.025)*P and  conclude that
    sg-ps-1 =the p-output of sg-ps-1
```

This rule is responsible to update the output of pressure sensor SG-PS-1

```
whenever the steam-outflow-rate F of sg-blr-
    1 receives a value then conclude that sg-
    fs-1 =F+random(-.025,.025)*F
```

This rule is responsible to update the output of flow rate transducer SG-FS-1

```
whenever the p-out P1 of sg-ps-1 receives a
    value then conclude that the p-output of
    sg-ps-2 =sg-ps-2 =P1 - random (0.9, 1.3)
```

This rule is responsible to update the output of pressure sensor SG-PS-2

```
for any pressure-sensor PS
   whenever the p-output of PS receives a
      value then
    conclude that PS= the p-output of PS
```

This rule is responsible to update pressure sensor reading

```
for any temperature-sensor TS
   whenever the t-output of TS receives a
      value then
    conclude that TS= the t-output of TS
```

This rule is responsible to update thermal couple reading

for any centrifugal-pump CP if the pump-open-state of CP = 1.0 then inform the operator that "Boiler Pump Is ON"	This rule is responsible to inform the operator the current state of centrifugal pump CP
for any centrifugal-pump CP if the pump-open-state of CP = 0.0 then inform the operator that "Boiler Pump Is OFF"	This rule is responsible to inform the operator the current state of centrifugal pump CP
if safe-boiler-pres = true then inform the operator that "Boiler is simulated under safe pressure"	This rule is responsible to inform the operator that the steam boiler is simulated under normal condition
if safe-boiler-pres = false then inform the operator that "Boiler is simulated under unsafe pressure"	This rule is responsible to inform the operator that the steam boiler is simulated under abnormal condition
if super-heater-on = true then inform the operator that "SuperHeaters are ON"	This rule is responsible to inform the operator that the super heaters is turned on
if super-heater-on = false then inform the operator that "SuperHeaters are OFF"	This rule is responsible to inform the operator that the super heaters is turned off
if city-water-on = true then inform the operator that "City Water has been turned on"	This rule is responsible to inform the operator that water source to condenser is turned on
if city-water-on = false then inform the operator that "City Water has been turned off"	This rule is responsible to inform the operator that water source to condenser is turned off

```
whenever blr-pressure-sp receives a value
    and when process-sim-active = true then
    conclude that state-update-simulate-time
    = 0 and conclude that state-update-
    simulate-on = true
```

This rule is responsible to keep the simultaneity of simulator clock

```
whenever p-dd receives a value then
    conclude that TREND-PRIMIT = trend-
    primit-calculate (p-d, p-dd) and conclude
    that trend-primit-num = trend-primit-
    calculate-num (p-d, p-dd)
```

This rule is responsible to calculate current primitive value according to the received FDD and SDD

```
whenever the source-total-flow F of city-
    water receives a value then conclude that
    sg-fs-2 = F + random (-0.025,0.025) * F
    and conclude that sg-fs-3 = F * random
    (0.32,0.36)
```

This rule is responsible to update the output of flow rate transducer of SG-FS-2 and SG-FS-3

```
whenever sensor-reading1 receives a value
    then conclude that the t-output of tc-1 =
    max(0,sensor-reading1 * 100)
```

This rule is responsible to update the output of thermal couple in ARSST device

```
whenever sensor-reading2 receives a value
    then conclude that the p-output of ps-1
    = max(0,sensor-reading1 * 100 + 63.8)
```

This rule is responsible to update the output of pressure transducer in ARSST device

APPENDIX E

Procedure to Set up Test Case in ARSST

GENERAL EQUPMENT DESCRIPTION:

The ARSST consist of three major components: ARSST containment vessel, ARSST control box and computer (A/D) board. The containment vessel houses test cell, the heater, and the thermocouple and insulation assembly.

GENERAL OPERATING PROCEDURES TO SET UP TEST CASE in ARSST:

1) Fill the test cell with sample and put the magnetic stirrer in the test cell if stirring is required during the experiment. The test cell is then insulated and put safely in the containment vessel.
2) The test cell is connected to heater cable wires.
3) Now insert the tip of thermocouple in the test cell. The position of TC so that tip is away from heater surface and below the mid plane of the test cell.
4) Fit the test cell extension tube and close the containment vessel.
5) Open the valve of extension tube and inject the accurately weighted sample mass in the test cell.
6) Close the extension tube valve.
7) Click on the ARSST setup software. Fill the required information of the test sample like sample name, mass, standard test volume and click on proceed to set up screen.
8) In the set up screen choose the required mode of heating. Depending on heating mode chosen, insert the values of heating rate required, auto-off heater criteria and data logging interval.

9) Click on the calibrate temperature and pressure. Normally at start of each experiment pressure calibration is required.

10) In pressure calibration, calibration of pressure from zero to highest possible pressure is required. The set A is normally referred to zero set pressure and set B refers to highest set pressure.

11) Open the gas inlet valve and slowly pressurerize the test cell up to the highest required pressure and click set B.(Here Nitrogen gas is used from the nitrogen cylinder)

12) Click on calibration completed and again you will back to set up screen.

13) Double click alarmer.exe file and wait until window being activated , which shows GSI is ready

14) Open the G2 software and load corresponding kb file

15) Click GSI Definition button on main workspace

16) Make the value of "sensor-interface" object change to 2 and click "Enable Sensor Input" button

17) Click ARSST button on main workspace and wait until the sensor output keep updating

18) Switch back to ARSST setup software and click on go to test screen.

19) Adjust the required initial nitrogen pressure by manipulating vent valve.

20) Click on start run and observe the temperature vs. time and pressure vs time profile.

21) Switch back to G2 software and open GDA interface