JEAS

JOURNAL OF ENGINEERING AND APPLIED SCIENCES

Journal of Engineering and Applied Sciences, Volume 16, Number 1, January 2020, 22-32

Development of Condition Based Self Diagnostic System for Manufacturing Industries

Nwandu E. E[1], Inyiama H. C[2], Nwalozie G. C[3]

[1]Department of Electronic & Computer Engineering, Nnamdi Azikiwe University Awka

[2]Department of Electronic & Computer Engineering, Nnamdi Azikiwe University Awka

[3]Department of Electronic & Computer Engineering, Nnamdi Azikiwe University Awka

[*]Corresponding Author's E-mail: ethelarubalueze@gmail.com

Abstract

In this work generic Multiprocessor Base Self-Diagnostic Process Control System has been developed. It features an Operator's console that facilitates easy fault identification and maintenance and is robust enough to simplify system attendance. In order to demonstrate how a process can be regulated through a multiprocessing self-diagnostic procedure, a soap production process for quality soap manufacture was used. The prototype was realized using homogenous multiprocessor system for different stages of the manufacturing process. Industrial thermocouples (sensors) and solid state relays were used to achieve the precision desired. A self-diagnostic algorithm was developed in C programming language using Mikro C Pro platform and implemented in the system. The current status of the product is a functional prototype which has been successfully tested in real production conditions. The system was able to monitor and control the temperature and duration of each of the various stages in the soap production process and present identified errors to the operator via a remote display with a wireless computer interface. Results obtained show that the system availability is 98.7% compared to about 70% before the multiprocessor system was introduced. This means that the system has very low failure rate and very low maintenance time because of the self-diagnostic features incorporated in it. Not more than one maintenance action was required per month and the quality of the output and effectiveness of the system is quite good.

Keywords: Diagnosis, multiprocessors, temperature control, sensors and soap manufacturing.

1. Introduction

Increased utilization of manufacturing systems is an ever present industrial challenge. Demands on improved productivity combined with reduced tied up capital and fewer employees are constantly increasing. All in order to match the global competition and survive as a world class company. One straightforward approach for increasing the production rate with the same or higher availability is to reduce system downtime, by implementing an effective fault diagnosis process. However, in industry today, fault diagnosis in automated manufacturing systems is mostly manual, highly empirical and error-prone procedure, which may vary widely from person to person and from shift to shift. Fault diagnosis in automated manufacturing systems, including both material processing and material handling with fast motions, is particularly hard because of the time critical synchronization between a number of control tasks and the production equipment.

As the complexity of a network increases, diagnosis of fault becomes a difficult task for network operators. Typically, one fault in the communication system produces large amount of alarm information, which is called alarm burst. Because of the huge information, manual cause identification becomes time consuming and error-prone. Therefore, automated fault diagnosis in computer networks is a problem. The consequences of faults in systems could be disastrous in terms of human mortality and environmental impact. To a less extent, fault detection in process and manufacturing industries is also crucial in order to improve production efficiency, quality of the product and cost of production. Current computer networks are becoming much larger and more complex. One single fault that occurs in one network component might cause considerably high volume of alarms to be reported to network operators, which is called alarm burst. Alarm burst may be as a result of fault reoccurrence, multiple invocations of a

service provided by a faulty component, generating multiple alarms by a device for a single fault. It can also be in the form of detection of and issuing a notification about the same network fault by many devices simultaneously, error propagation to other network devices causing them to fail and, as a result, generate additional alarms. Thus, it is a challenge for network operators to quickly and correctly identify the root cause, by analyzing those large amounts of alarms arising from these possible causes.

Due to the complexity of computing systems and difficulty of formalizing the scope of the diagnosis task itself, diagnosis has historically been a large manual process requiring significant human input. However, techniques to automate as much of the process as possible have significantly grown in importance. In domains such as communication networks and Internet services, the sheer scale of modern systems and the high volumes of impairments they face drive such trends. While in domains such as embedded systems and spacecraft, its increasing complexity together with the need for autonomic operation (i:e:, selfhealing) when human expertise is not available, that are the drivers (Agarawala et al 2007). Due to the diversity of the domains, a variety of failure diagnosis techniques drawing from diverse areas of computing and mathematics such as artificial intelligence, machine learning, statistics, stochastic modeling, Bayesian inference, rule-based inference, information theory, and graph theory have been studied in the literature. Finally, when automated techniques fail, approaches that assist humans perform diagnosis more efficiently via the use of visualization aides have also been widely deployed.

System-level diagnosis is a technique for fault tolerance that strives to identify the faulty elements in a system. This is done by deduction, based on information in the form of results of tests applied to the elements. Once the faulty elements have been identified, the system is able to isolate them, ignore their output and to initiate a repair operation such that the reliability of the system can be maintained in the long run. Ultimately the expected reliability of a system depends on what happens when the system behaves inappropriately. If lives of humans or large sums of money are at stake, then the system must be very reliable. Examples of ultra-reliable multiprocessor systems include the control systems for an airplane, a space ship or a nuclear reactor. Other systems, such as automated bank tellers or airline reservation systems, can be out of service for small periods of time without harm. Nonetheless the distributed information in these systems is expected to remain consistent despite any down time that may occur. The more complex a system is, the more difficult it is to achieve reliability. Fortunately multiprocessor systems have built-in redundancy. The multitude of processors and the equipment that allows connections between them can allow the system to perform more work than a single-processor system. More importantly, it is also this redundancy of equipment that enables the system to continue to function as needed, though one or more of the processors have stopped working correctly or there are problems in the network connecting the processors.

1.1 Literature Survey

Cherkasova et al (Cherkasova et al 2008) use queues to model the relationship between CPU usage and transaction response times for a transaction mix. They also exploited regression to define an application performance signature that allows them to detect software upgrades by monitoring changes in the application signature. Stewart, Kelly and Zhang, (2007) model the relationship between multiple physical resources, namely CPU, disk and network, and response times for a transaction mix. These models need to be re-trained to cope with new transaction types. They also ignored interaction effects across transaction types and implicitly assume that queuing is the only manifestation of congestion.

Sherlock (Bahl et al 2007) and Khanna et al. (Khanna et al 2007) extended on Shrink and SCORE to deal with multi-level dependencies and with more complex operators that capture load-balancing and failover mechanisms. These techniques infer the root-cause by computing the probability that errors propagate from a set of possible root-cause nodes to the observation nodes. They indicted the root-cause nodes that best explain the symptoms at the observation nodes, and scale by assuming that there can only be a small number of concurrent problems in the system at a given time. Rish et al. (2004) proposed an active probing approach that exploits a dependency matrix to represent the failed components that each probe, e.g. server ping, detects. Active probing allows probes to be selected and sent on-demand, in response to one's belief about the state of the system. At each step the most informative next probe was computed and sent. As probe results were received, belief about the system state is updated using probabilistic inference. This process continues until the problem was diagnosed. They extended their active probing approach to cope with dynamic systems presented by Rish et al (2005), where problems may occur and disappear, by maintaining two sets of probes: one set for repair detection to monitor nodes that are known to have failed, and another set for failure detection to monitor nodes that are known to be working. Their approach assumes a sequential fault model in which only one fault or repair can occur at a time. Joshi et al. (2005) used a

Bayesian approach to diagnose problems in systems with different types of monitors, or probes that have differing coverage and specificity characteristics. They use a dependency matrix to represent the probability that a monitor detects a failure in a component, and incrementally update their belief about the set of failed components based on the observed monitor output.

Agarwal et al (2006) used change-point detection and problem signatures to detect performance problems in enterprise systems. They detect abrupt changes in system behavior by monitoring changes to the mean value of performance counters over consecutive windows of time. This technique does not scale well if the number of nodes and metrics is large. NetMedic developed by Kandula et al (2009) diagnosed propagating problems in enterprise systems by analyzing dependencies between nodes, and correlations in state perturbations across processes to localize problems. NetMedic represents state for each system component as a vector that indicates whether each metric was anomalous or normal by assuming that each metric obeys a normal distribution and flagging anomalies based on deviation from the mean. If two components which depend on each other are anomalous, NetMedic searches for time periods where the source component's state is similar to its current state, and searches for destination states that have experienced significant changes in the same period. These destination states are the likely culprits.

Draco developed by Kavulya et al (2011) performed statistical diagnosis of problems in large Voice-over-IP (VoIP) systems by comparing differences in the distributions of attributes, such as hostnames and customer IP addresses, in successful and failed calls. Draco assumed that these attributes were drawn from a Beta distribution and localized problems by identifying attributes that were most correlated with failed calls. By comparing successes and failures over the same window of time, Draco avoided the need for separate learning passes, and can thus diagnosed problems that have never been seen before.

Distributed architecture for monitoring and diagnosis (DIAMOND) system architecture is a set of distributed cooperating tasks. Each task was associated with a specialized agent, namely the monitoring agent, which was interfaced to the industrial application, a set of diagnostic agents to identify the functional state of the plant, a conflict resolution agent to investigate whether the diagnostic results were contradicting or complementing each other, a facilitator agent to manage networking and mediating between different agents, a blackboard agent for storing the diagnoses, and a user interface agent for presenting the results to the operator (Worn 2004). The DIAMOND system was implemented using the KQML-CORBA- (Knowledge Query and Manipulation Language) based architecture, in which the different agents were implemented as distributed CORBA objects. The system prototype was evaluated while monitoring and diagnosing the water stream cycle chemistry of a coal-fired power plant (Worn 2004).

2.0 Material and methods

In this work a multiprocessor based self-diagnostic process control system for quality soap production was developed. The operation of this system was automated and computerized. The temperature of the various stages of soap making process was controlled through the signal obtained from the thermocouple, embedded in the system which sends microvolt signals to the computer. The signal was amplified through the micro controller program. Controlled temperature was needed for each stage of the process machine, this was achieved by using a single microprocessor to monitor and control each of the various stages. These microprocessors would report their status to a central microprocessor that would process the data and store it in a database and equally perform the diagnostic and effecting control using the data. To produce quality soap, every section of the continuous process plant has its own different temperature value range. The final soap quality considered in this work was the soap physical appearance or the soap hardness (texture) assuming that accurate chemical variables and compositions are used**.**

2.1 Mathematical Modeling of the System

In the temperature controlled soap quality model presented, two major factors; temperature and duration of heating or cooling are used to control the quality of the finished soap. As shown in Figure 2, let:
S_R = raw soap properties,
S_s = saponified soap properties
t_s = saponification time
Θ_1 = saponification heating rate
S_c = chilled soap properties
t_c = chilling time

$\emptyset_1$ = chilling rate
S_m = milled soap properties
t_m = milling time
S_p = plodded soap properties
t_p = plodding time
$\emptyset_3$ = plodding rate of cooling
S = finished soap percentage characteristic
t_n = duration of time at nose cone
θ_2 = nose cone rate of heating
Generally, using Fourier's law of heat conduction for one-dimensional heat conduction equation,

$$Q = -kA\frac{\partial T}{\partial x} = \rho C\frac{\partial T}{\partial t} \quad (1)$$

Where, k, A, ρ, and C are the properties of the material and are given as:
Q = rate of change of energy content (heat rate).
k = inverse of specific heat of the material
A = thermal conductivity of the material
ρ = density of material
C = specific heat capacity of the material
T = temperature
t = time
From equation (1), it implies that:

$$T = \frac{1}{\rho C}\int_{t_1}^{t_2} Q dt \quad (2)$$

Using figure 2 and applying equation (2), then it becomes that:

$$S_s = S_R \int_{t_0}^{t_s} \theta_1 \, dt \quad (3)$$

$$S_c = S_s \int_{t_{c1}}^{t_{c2}} \emptyset_1 \, dt \quad (4)$$

Putting equation (3) into (4)

$$S_c = \left(S_R \int_{t_0}^{t_s} \theta_1 \, dt\right) \int_{t_{c1}}^{t_{c2}} \emptyset_1 \, dt = \int_{t_0}^{t_s} \theta_1 \, dt \int_{t_{c1}}^{t_{c2}} \emptyset_1 \, dt \quad (5)$$

Again,

$$S_m = S_c \int_{t_{m1}}^{t_{m2}} \emptyset_1 \, dt \quad (6)$$

Substituting for Sc in equation (6),

$$S_m = \left(S_R \int_{t_0}^{t_s} \theta_1 \, dt \int_{t_{c1}}^{t_{c2}} \emptyset_1 \, dt\right) \int_{t_{m1}}^{t_{m2}} \emptyset_1 \, dt$$

$$= S_R \int_{t_0}^{t_s} \theta_1 \, dt \int_{t_{c1}}^{t_{c2}} \emptyset_1 \, dt \int_{t_{m1}}^{t_{m2}} \emptyset_1 \, dt \quad (7)$$

Also, we have that:

$$S_p = S_m \int_{t_{p1}}^{t_{p2}} \emptyset_3 \, dt \quad (8)$$

Putting equation (7) into equation (8), we have:

$$S_p = \left(S_R \int_{t_0}^{t_s} \theta_1 \, dt \int_{t_{c1}}^{t_{c2}} \emptyset_1 \, dt \int_{t_{m1}}^{t_{m2}} \emptyset_1 \, dt\right) \int_{t_{p1}}^{t_{p2}} \emptyset_3 \, dt$$

$$= S_R \int_{t_0}^{t_s} \theta_1 \, dt \int_{t_{c1}}^{t_{c2}} \emptyset_1 \, dt \int_{t_{m1}}^{t_{m2}} \emptyset_2 \, dt \int_{t_{p1}}^{t_{p2}} \emptyset_3 \, dt \quad (9)$$

Now,

$$S = S_p \int_{t_{n1}}^{t_{n2}} \theta_2 \, dt \quad (10)$$

Substituting equation (9) into equation (10) therefore,

$$S = \left(S_R \int_{t_0}^{t_s} \theta_1 \, dt \int_{t_{c1}}^{t_{c2}} \emptyset_1 \, dt \int_{t_{m1}}^{t_{m2}} \emptyset_2 \, dt \int_{t_{p1}}^{t_{p2}} \emptyset_3 \, dt \right) \int_{t_{n1}}^{t_{n2}} \theta_2 \, dt$$

$$= \int_{t_0}^{t_s} \theta_1 \, dt \int_{t_{c1}}^{t_{c2}} \emptyset_1 \, dt \int_{t_{m1}}^{t_{m2}} \emptyset_2 \, dt \int_{t_{p1}}^{t_{p2}} \emptyset_3 \, dt \int_{t_{n1}}^{t_{n2}} \theta_2 \, dt$$

$$S = S_R \left(\int_{t_0}^{t_s} \theta_1 \int_{t_{c1}}^{t_{c2}} \emptyset_1 \int_{t_{m1}}^{t_{m2}} \emptyset_2 \int_{t_{p1}}^{t_{p2}} \emptyset_3 \int_{t_{n1}}^{t_{n2}} \theta_2 \right) dt \qquad (11)$$

By analogy with equation (2), $S_R = \frac{1}{\rho C}$. Therefore,

$$S = \frac{1}{\rho C} \left(\int_{t_0}^{t_s} \theta_1 \int_{t_{c1}}^{t_{c2}} \emptyset_1 \int_{t_{m1}}^{t_{m2}} \emptyset_2 \int_{t_{p1}}^{t_{p2}} \emptyset_3 \int_{t_{n1}}^{t_{n2}} \theta_2 \right) dt \qquad (12)$$

Equation (12) could therefore be used to characterize the quality of soap to be produce. Adjusting the heating rate at each stage of production will help determine the required temperature at the various stages as well as that of the finished soap. To do this, the soap content temperature must be monitored with time for effective control, using a single microprocessor to monitor and control each of the various stages. These microprocessors would report their status to a central microprocessor that would process the data and store it in a database and equally perform the diagnostic and effecting control using the data.

2.2 Diagnostic Procedure

From the model developed when a failure or an error occurs in the system, the microprocessor attached in that stage would identify the failure and processes the failure before informing the central control unit of the type of failure or error. The central control unit would inform and instruct the microprocessor based on the standard operating condition available on its memory, to carry certain specified actions and when that is done and the system is restored, the central control unit will append a corresponding action code to the action that resolved the identified failure. The central control unit would communicate this operation to the base station that would help the system to learn the failures and the action taken to resolve them and stores the same information on the database. This information was used to develop a diagnosis algorithm using "IF-Then rule" that would help the system to identify failures or errors in the future that have occurred in the past (using the error code) and immediately apply that specific action (using the action code) that resolved the failure then without any human intervention.

2.3 Failure Detection

As soon as failure are detected the knowledge base is notified and data concerning its duration and number of occurrences so far are recorded and transmitted. Detectable failures are either caused by the increase or decrease in temperature of a particular stage or a stage exceeding its expected duration. When a detectable failure occurs on the production system, the central processor increments its counter for that particular failure and records its duration. Once the alarm is acquitted, the information collected previously is sent to the knowledge base for determining the failure's frequency. Some of the detectable failures (which are functions of temperature and time) in the soap manufacturing system are listed below;

Clogging in the machine
Breakup of soap noddles
Milling valve gap (roughness and grit)
Soap sticking in the plodder barrel
Gramage inconsistency
Poor texture of the soap
Soap sticks on ejection actuator

2.4 Diagnosis

The diagnosis method adopted in this work is the condition based self-diagnosis. This is based on the principle of monitoring the condition of machinery and repairing it just prior to failure or an unacceptable level of performance degradation. The diagnostics program uses the failure tree to make a list with the components that have the highest probability to have failed. For the failures already treated in the past, "if … then" rules were used and combined to failure trees to determine the cause of the failure and their associated maintenance procedure which was then stored

in the knowledge base with an action code. If after this step no cause for the failure is found, then a manual diagnosis method is initiated which will guide the technician to determine the failed component(s). Once the repairs are made, the operator creates an intervention report which contains the procedure he used or created to eliminate the failure. Then from this report an "if … then" rule is created and stored in the knowledge base for future usage.

3.0 Results and Discussions

The implementation and testing of the condition based self-diagnostic system for quality manufacturing was carried out on a continuous soap manufacturing plant. The soap manufacturing belongs to Chidioka & Sons Ltd, a local soap manufacturing company in Onitsha, Onitsha south local government area of Anambra State, South East Nigeria. The developed model was used to monitor and control the quality of the soap produced and the setup is as shown in figure 1.

In the implementation of the multiprocessors based self-diagnostic system, the various microprocessors attached to the various units of the soap manufacturing machine would take the time and temperature readings of their respective stages continuously and compare the readings with a preset temperature range for that stage. These microprocessors report their current status periodically to the central processing unit till the end of the production, the central processing unit would then display the information and stores the same in the database. In the course of taken these temperature readings if an error or failure is discovered in any of the stage, the corresponding microprocessor will attend to the error and report both the error and action taken to resolve the error to the central processing unit using the error codes and action codes. But when an error that cannot be handled by the unit microprocessor occurs, it would append a new error code and report immediately to the central processing unit that would then alert the operator with the information about the unit where there was discovered error. The operator would attend to the error and through the attached keypad enter into the PC the action taken to resolve the error and the central processing unit would assign a new action code to the solution for future use and stores same in the database of the system.

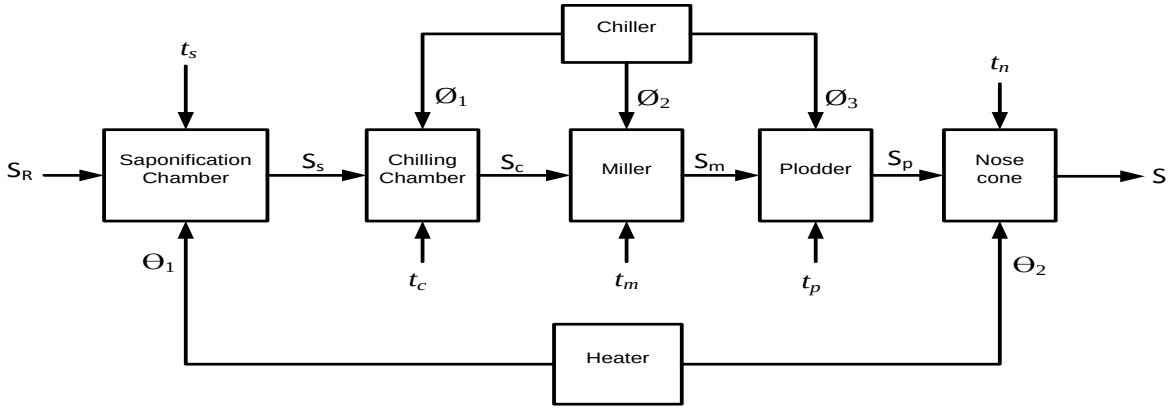

Figure 1: Block Diagram of the Automated Soap Manufacturing Process

3.1 PC-Side Graphical User Interface (GUI)

A PC-side graphical user interface was developed to make interaction with multiprocessor based system possible via a computer. This application defines a graphical user interface (GUI) through which users can feed data, send commands for data acquisition and as well view resolved failures. The level and extent of this application was designed according to the need of the organization and could be developed in any high level software language The GUI as shown in figure 2 was developed by Microsoft Visual Studio IDE and written in Visual Basic.NET.

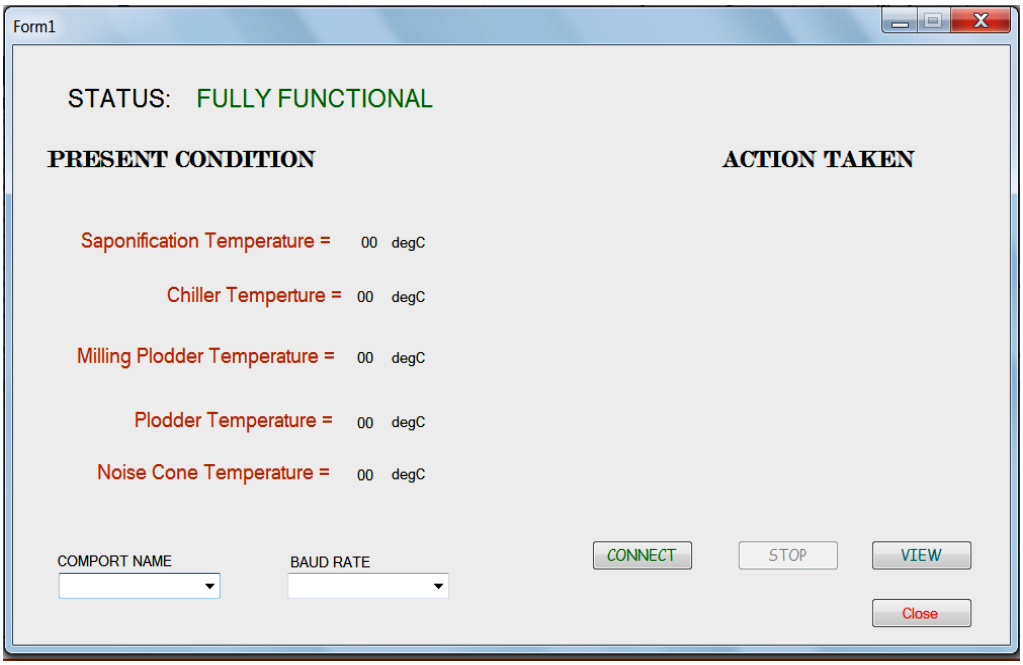

Figure 2: Screen shot of the GUI for data acquisition

3.2 Diagnosis Algorithm Development

When a failure or an error occurs in the system, the microprocessor attached in that stage identified the failure and process the failure (by appending a specific error code) before informing the central control unit of the type of failure or error. The central control unit then informed and instructed the microprocessor based on the standard operating condition available on its memory, to carry certain specified actions and when that is done and the system is restored, the central control unit then appended a corresponding action code to the action that resolved the identified failure. The central control unit would communicate this operation to the base station that would help the system to learn the failures and the action taken to resolve them and stores the same information on the database. This information was used to develop a diagnosis algorithm using "IF-Then rule" that would help the system to identify failures or errors in the future that have occurred in the past (using the error code) and immediately apply that specific action (using the action code) that resolved the failure then without any human intervention. If new failures or errors are identified in the future the microprocessor attached to that unit will append new error code to it and notify the central control which in turn will alert the operator using the Buzzer system. Then the operator would check the system to restore the system and afterwards saves the action taken in the system which would append new action code to the action. This would help the system to learn of the new identified and resolved failure for future operations. The pseudo code is given hereunder.

1. *Start.*
2. *Initialize.*
3. *Start timer.*
4. *Receive temperature readings from all five chambers.*
5. *Compare temperature readings with predefined readings for error.*
6. *If error exist then,*
7. *Check database for its error code.*
8. *Else, move down to number 14.*
9. *If error code is present in the database then,*
10. *Use corresponding action code to resolve the error.*
11. *Check if error is resolved.*
12. *If yes then,*
13. *Update database and close database.*
14. *Display system status, temperature details and actions taken.*

15. *Else, alert user or operator using display and sound.*
16. *Operator enters details of new error in database.*
17. *Return to number 2.*

End

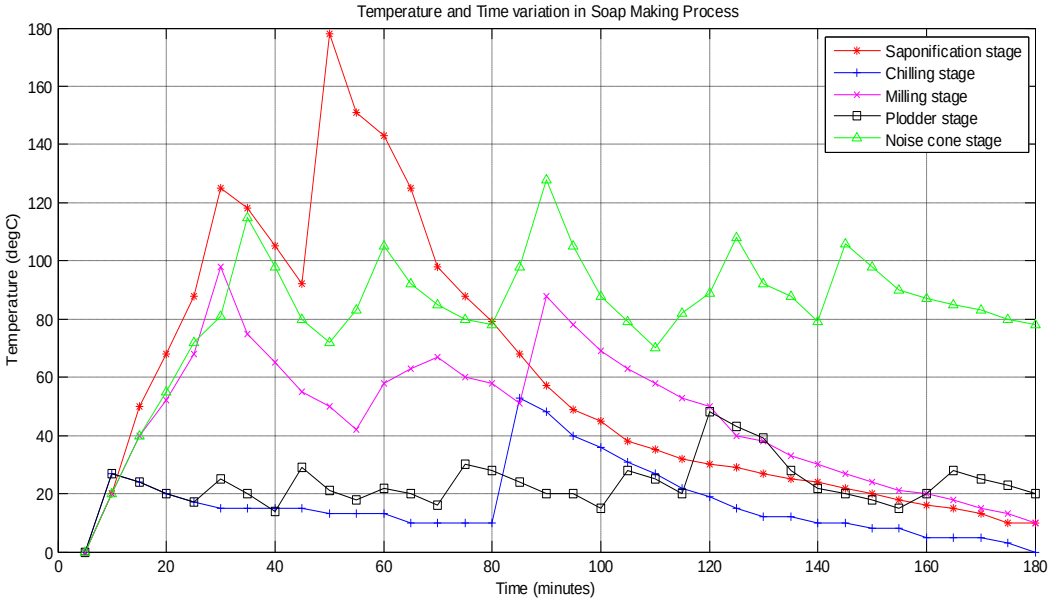

Figure 3: Temperature and Time relationship in soap production process

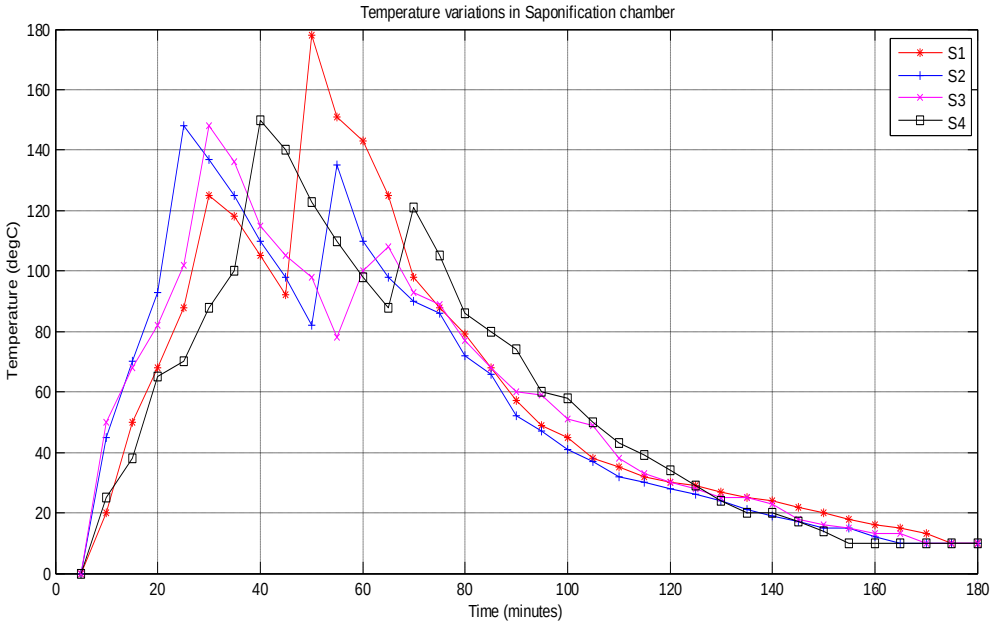

Figure 4: Average temperature variation in the saponification chamber

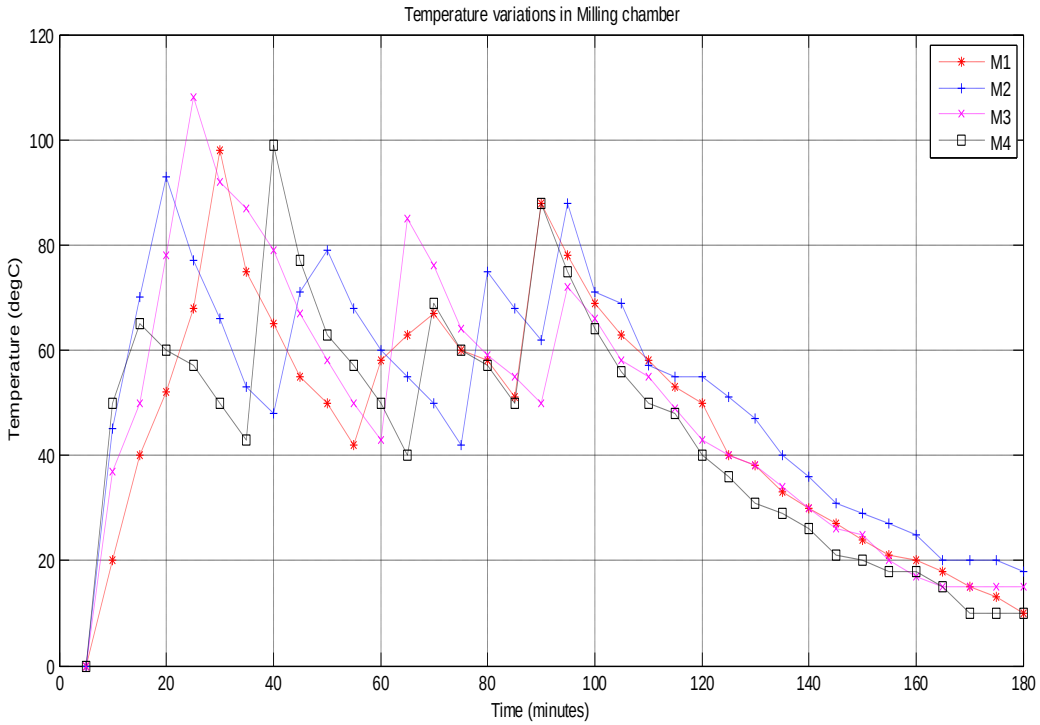

Figure 5: Average temperature variation in the Milling chamber

Figures 3, 4, and 5 show the results obtained during the testing of the developed model using the local soap manufacturing company. During the testing of the designed system, which involved the complete production of soap from the raw material to the finished soap. It was discovered that the system monitored and controlled the soap manufacturing machine and ensured that the quality of soap produced met the stated expectations, as shown in figure 3 where the self-diagnostic system was able to diagnose and self-corrected the soap manufacturing operations and returned the operating conditions to the standard stated and/or observed conditions. Similarly, figures 4 and 5 show the fluctuations and corrections of the operating conditions in some of the chambers of the continuous soap manufacturing plant. Equally, there were some discovered errors that required the attention of the operator and the solution rendered was also recorded. The entire process was completed normally on average of about 3 hours.

3.3 Evaluation of the Diagnosis Algorithm

The performance of specific detection and diagnostic algorithm or subsystems of a CBM system are measured with Performance Metrics. The functionality of the diagnostic algorithm or subsystems directly contributes to the overall effectiveness of the entire system. However, the ability to assess the accuracy and robustness of particular algorithms is often more straightforward when the technologies making up the system are checked separately. Also, from a design and development point of view, it is often more logical to work on the improvements to specific algorithms or processes at the elemental level rather than the overall systems level.

Metrics of performance for diagnostic/prognostic algorithms or subsystems are arranged into three categories;

- ✓ Detection,
- ✓ Isolation,
- ✓ Prognosis.

Detection metrics measure the ability of diagnostic tools to correctly classify production operation as either normal or anomalous. Isolation metrics measure the ability of diagnostic tools to accurately identify the root cause and corrective action for a fault. Prognosis metrics measure the ability of prognostic systems to accurately forecast the future condition of a production system. Scores from the individual performance metrics are combined according the hierarchy to produce summary scores for each category, and for overall performance. The ability of diagnostic system to detect and isolate faults or to predict failures is measured as a function of the fault severity. Fault severity

must be established by objective and irrefutable measures to ensure that the assessments based upon it are accurate and impartial. In this work, a set of experiments and an experimental analysis method (using multiple regression analysis) were used to test the significance of the experimentally controlled parameters in producing quality of soap. One subset of parameters which are identified as potentially affecting soap quality consists of heating rate, cooling rate and duration of each stage. These significant parameters that affect the quality of the soap are first established within a feasible parameter space and are given in Table 1.

Table 1: Summary of the variables for the soap

Stage Parameter	Coded level of Variable	
	Low	**High**
Heating rate θ (Deg C)	50	105
Cooling rate $\emptyset$ (Deg C)	0	25
Duration t (mins)	0	60

4.0. Conclusion

A condition based fault diagnosis process control system was designed for quality manufacturing. This work used a continuous soap manufacturing plant as an example of an industrial machine using sensor signals along with methods and algorithms. The approach is based on sensor readings and a relevant feature identification and extraction process based on those sensor signals. The approach is mainly based on the condition based monitoring methodology and it enables the collection of valuable sensor data from machine on a regular basis for use in fault diagnosis and for storage for future use. Evaluations have shown that the proposed approach has been proven successful and reliable in diagnosing faults in the soap manufacturing machine.

References

Agarwal .M.K, Gupta .M, Mann .V, Sachindran .N, Anerousis .N, and Mummert .L.B, 2006.Problem determination in enterprise middleware systems using change point correlation of time series data. In IEEE/IFIP Network Operations and Management Symposium, pages 471-482, Vancouver, Canada.

Agarwala .S, Alegre .F, Schwan .K, and Mehalingham .J, 2007. E2eprof: Automated end-to-end performance management for enterprise systems. In IEEE Conference on Dependable Systems and Networks, pages 749-758,

Bahl .P, Chandra .R, Greenberg .A, Kandula .S, Maltz .D, and Zhang .M, 2007. Towards highly reliable enterprise network services via inference of multi-level dependencies. In ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), pages 13-24, Kyoto, Japan.

Cherkasova .L, Ozonat .K, Mi .N, Symons .J, and Smirni .E, 2008. Anomaly? Application change? or workload change? Towards automated detection of application performance anomaly and change. In IEEE Conference on Dependable Systems and Networks, pages 452-461, Anchorage, Alaska.

Joshi .K, Sanders .W, Hiltunen .M, and Schlichting D, 2005. Automatic model-driven recovery in distributed systems. In IEEE Symposium on Reliable Distributed Systems, pages 25-38, Orlando, Florida

Kandula .S, Katabi .D, and Vasseur .J.P, 2005. Shrink: A Tool for Failure Diagnosis in IP Networks. In ACM SIGCOMM Workshop on mining network data (MineNet-05), Philadelphia, PA.

Kandula .S, Mahajan .R, Verkaik .P, Agarwal .S, Padhye .J, and Bahl .P, 2009. Detailed diagnosis in enterprise networks. In ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), pages 243-254, Barcelona, Spain.

Khanna .G, Laguna .I, Arshad .F, and Bagchi .S, 2007. Distributed diagnosis of failures in a three tier e-commerce system. In IEEE Symposium on Reliable Distributed Systems, pages 185-198, Beijing, China.

Kavulya .S, Joshi .K, Hiltunen .M, Daniels .S, Gandhi .R, and Narasimhan .P, 2011. Practical experiences with chronic discovery in large telecommunications systems. In ACM Workshop on Managing Systems via Log Analysis and Machine Learning Techniques (SLAML), Cascais, Portugal, October.

Qeethara K. S, 2011. Artificial Neural Networks in Medical Diagnosis. IJCSI International Journal of Computer Science Issues, vol. 8, issue 2, pp 150-154.

Rish .I, Brodie .M, Ma .S, Odintsova .N, Beygelzimer .A, Grabarnik G, and Hernandez .K, 2005. Adaptive diagnosis in distributed systems. IEEE Transactions on Neural Networks, 16(5):1088-1109.

Rish .I, Brodie .M, Odintsova .N, Ma .S, and Grabarnik .G, (2004). Real-time problem determination in distributed systems using active probing. In IEEE/IFIP Network Operations and Management Symposium, pages 133-146, Seoul, South Korea.

Stewart .C, Kelly .T, and Zhang .A, 2007. Exploiting nonstationarity for performance prediction. In European conference on Computer systems (EuroSys), pages 31-44, Lisbon, Portugal.

Worn, H. (2004). DIAMOND: Distributed multi-agent architecture for monitoring and diagnosis, journal of Production Planning and Control 15: 189–200.