A SYSTEM MONITORING MODEL BY EXAMINING ENTITY DYNAMICS

WANG LEI

Ph.D

The Hong Kong Polytechnic University

2014

The Hong Kong Polytechnic University

Department of Industrial and Systems Engineering

A System Monitoring Model by Examining Entity Dynamics

WANG Lei

A thesis submitted in partial fulfilment of the requirements for

the degree of Doctor of Philosophy

August 2013

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____(Signed)

WANG Lei (Name of student)

ABSTRACT

Monitoring plays an essential role in the manufacturing processes, as it bridges the gap between system conditions and the necessary corrective activities. Generally, for a Computer Integrated Manufacturing (CIM) system, monitoring is about collecting various facility signals, which are fed into specific models to interpret the signals. However, from the managerial viewpoint, the healthiness of the whole system is more desirable than the operation detail. In effect, the monitoring operation of a CIM system is analogous to diagnosing a system by checking the functioning of some targeted domains. Inspired by the distinctive philosophy that a proper system should be working in complete harmony, a novel method for presenting a holistic picture of a manufacturing system by examining the flowing entities is presented in this research.

This research was conducted in three stages. First, a manufacturing system was modelled as the integration of a set of *Regions of Interest* (ROIs) in a high level manner. Second, analogous to the concept of checking blood pulses in the human body, several features were extracted from a system to constitute the "pulses" of an ROI; these features include the *Regional Inconsistency* (RI), the *Inter-component Arrival Time* (IAT), the *Inter-component Leaving Time* (ILT), and the *Instant Work-In-Progress* (IWIP). A reasoning scheme was then established to detect two types of popular abnormalities (*blockings* and *slowdowns*) in an ROI. Third, an ROI segmentation technique was developed to assist in the design of the monitoring framework by taking into consideration the tolerable system response time.

At the outset of the study, it was anticipated that, based on analyzing the “pulses” tones of all ROIs involved, the healthiness of the holistic system could be reflected. Simulation experiments were conducted to validate the effectiveness of the monitoring approach proposed in this research. It was found that, in terms of the hardware requirements, only simple counter devices with time-stamp functions are needed, and this highly enhances the portability of the proposed approach.

PUBLICATIONS ARISING FROM THE RESEARCH

Journals

Wang, L. and Chan, C.Y. (2012). Formulation of novel production line monitoring technique. *International Journal of Production Research,* DOI: 10.1080/00207543.2012.662304

Wang, L. and Chan, C.Y. (2013). Design of Counter-based Production Monitoring Technique Subject to Response Time Constraint. *Engineering Letters, (Accept with minor revision*)

Wang, L. and Chan, C.Y. (2013). Development of a Low-cost Indirect Monitoring Method to Oversee Production Line. *Computers & Industrial Engineering, (*Under review)

Conference

Wang, L. and Chan, C.Y., Establishment of a low cost production monitoring technique, Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering 2012, WCE 2012, 4-6 July, 2012, London, U.K., pp1391-1394

ACKNOWLEDGEMENT

My deepest gratitude goes, first and foremost, to my chief supervisor, Dr. C.Y. Chan, for his great patience and considerable guidance throughout the research. This thesis would not have been possible without his moral support and continuous encouragement. His innovative thinking and serious attitude to research have set a good example for my future professional development. I would also like to express my heartfelt gratitude to Professor H.C. Man, who has instructed and helped me a lot in the past three years, with both academic study and life in Hong Kong.

My thanks go to my beloved family for their categorical support and great confidence in me all through these years. Special thanks should be given to my wife Sophie, for her understanding, trust, continual support, and for accompanying me through this period of time.

Finally, I sincerely appreciate the Department of Industrial and Systems Engineering of The Hong Kong Polytechnic University for providing a grant to support this research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter One - Introduction to dynamic system monitoring

The days when Henry Ford was advertising that "the public can have a Model T Ford in any colour desired, as long as it is black" are gone forever. Aside from their individual preferences, contemporary customers consider many criteria when picking suppliers, including quality of product, cost, service, and delivery time (Rehg & Kraebber, 2005). To satisfy the increasingly rigorous requirements and to advance in the "survival of the fittest" market, many modern manufacturers are rushing to introduce advanced technologies into the manufacturing process in order to enhance their competitive capabilities.

Despite rapid technological improvements in manufacturing systems, the process generally involves the transition from raw materials to products or semi-products through a series of machining procedures. The process should always be supervised and some adjustments are required to keep it on track. System monitoring, as a bridge connecting current actuation and subsequent control, has also received continuous attention from both the academic and industrial areas. Typically, the system relies on diversified sensors to collect feedback information; the integration of the information plus general knowledge about the system constitutes the input of managerial activities such as fault detection, failure analysis, and process control.

The potential approaches to supervising a system are determined by the angle from which to look at it and the purpose for which to conduct it. Generally speaking, the monitoring approaches within a manufacturing system can be classified roughly into two

categories, the western Computer-Integrated Manufacturing (CIM) monitoring philosophy and the Japanese Toyota Production System monitoring philosophy.

1.1 Introduction to Computer-Integrated Manufacturing system monitoring

The concept of Computer-Integrated Manufacturing (CIM) emerged in the 1980s in the light of the fast developments occurring in electronic and computer technologies (Koren, 1983). By integrating three key factors, person, technology, and running management, in the production processes to collect the system information to regulate the operations, CIM tends to optimize the output from all of these aspects (Masood & Khan, 2004). When it comes to the implementation of CIM in a practical situation, a pyramid style functional hierarchy is generally adopted, as in Figure 1-1 (Sandoval, 1994). At the bottom of the pyramid is the enterprise infrastructure; this can be a production plant or a shop floor, in which raw materials are transformed to products, where various sensory devices are employed to collect signals from work-stations or processes. In the middle of the pyramid, all resources are involved in channelling goals from the upper level to the lower level in an effective manner, and reversible feedback execution status is indispensable. Finally, the peak of the pyramid is associated with holistic strategy and management objectives coming from the decision-makers through information flow.

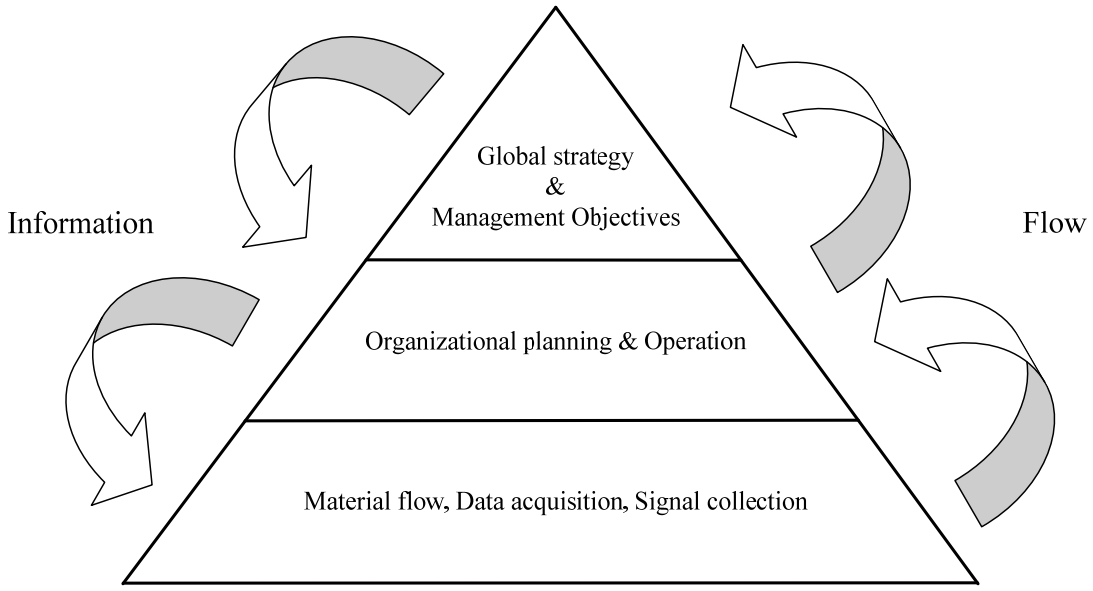


Figure 1-1 CIM pyramid style functional hierarchy (Sandoval, 1994)

Successful CIM cases have been presented in many previous studies. For instance, Gunasekaran, Marri and Lee (2000) proposed a framework for Small and Medium-Sized Enterprises (SMEs), and described its implementation in a medium-sized company with about 200 employees and an annual turnover of £8m. The issue of how to apply CIM successfully in industrial enterprises has also been discussed (Jardim-Goncalves et al., 1997), and to improve the success rate, some hybrid models have been proposed to assist decision-makers in selecting appropriate CIM alternatives before investing (Yurdakul, 2004). Despite the obvious effectiveness of CIM (Al-Ahmari, 2007; Nagalingam & Lin, 2008), there are several inherent disadvantages. First, the implementation process tends to be rather tough due to incompatibilities among facilities (Gourgand, Lacomme, & Traore, 2003), and the results are not always satisfactory (Mcgaughey & Roach, 1997). Based on the surveys of both academic researchers and practitioners, apart from the human-related

subjective impacts, topping of the list of obstacles to a successful CIM are generally poor system compatibility, lack of specialized knowledge and technical skills to the whole system, and the expensive and time involved in building a system model (McGaughey & Snyder, 1994; Mcgaughey & Roach, 1997).

In fact, the enhancement of manufacturing ability has led to a significant increase in the importance of production intelligence, as the successful operation of a CIM system requires not only great efficiency but also high reliability. This keeps driving researchers to conduct more work on the system monitoring aspect and also on fault diagnosis. Presently, most CIM systems rely on heterogeneous sensors to collect signals from facilities/processes. The collected signals are processed and used as input to a dedicated model to reflect the system status. Typically, the establishment of a CIM monitoring system involves three steps. The first step is to generate a model and define informative vectors to reflect system situations; the second is to search, purchase, and deploy different sensors based on the informative vectors; and the last step is to work out appropriate schemes to supervise and diagnose the system by inputting the integrated information into the model and coordinating the operations/inter-communications among devices.

In reality, the design and implementation of an efficient CIM system is a big challenge, which demands elaborative consideration of not only the hardware aspects but also the software. As demonstrated by Gourgand et al. (2003), the practical workload on the latter part actually exceeded seventy percent of the entire design. In fact, one of the most challenging issues for the success of a CIM system is the establishment of a

coherent system-monitoring framework through the proper coordination of the various computer-based facilities.

To sum up, existing CIM monitoring approaches have several common deficiencies. First, the model is usually tailored to a specialized domain, which leads to poor adaptability and is difficult to maintain. Unfortunately, contemporary enterprises are continually being reminded to adapt to the rapidly evolving markets so as to survive in the fierce competition (Zaremba & Prasad, 1994). This conflict brings great challenges to the design of modern CIM systems and also to the monitoring work. Second, the requirement of numerous sensor devices and related facilities to collect signals from the processes put quite a cost burden on an enterprise, especially an SME. Third, it is usually complicated to coordinate various signals and requires a lot of effort to manage the whole system. As a matter of fact, to a certain extent, the major problem in decision making is not information insufficiency but, quite the contrary, information overload (Edmunds & Morris, 2000).

1.2 Introduction to Toyota Production System monitoring philosophy

Unlike most western enterprises, which are continuously pursuing integration techniques for monitoring the manufacturing floor, Japanese companies adopt a disparate manufacturing philosophy. The Toyota Motor Corporation invented the Toyota Production System (TPS) in the last century. The ultimate goal of the TPS is to reduce waste. Two of its major components are Just-In-Time (JIT) and Jidoka. JIT means producing the right quantity of products at the right time, thereby reducing the inventory and associated holding costs. Jidoka is the defect-prevention solution to timely recognition of systematic problems, with which the processing line reliability can be

guaranteed (Amasaka, 2009). By focusing on processes with an eye towards the quality outcome, Toyota has long been recognized as a leader in the automotive manufacturing industry (Liker, 2003).

Despite the successful application of the TPS philosophy within Toyota and the overwhelming influence on the global manufacturing industry (George et al., 2011; Dombrowski et al., 2010; Hunter, 2008), the success of TPS spread over national boundaries seems not to be as smooth as the concept itself (Nakamura, Sakakibara, & Schroeder, 1996). The possible reasons lie in the socio-economic context, including differences in cultures, business models, and social conditions (Lee & Jo, 2007). Liker (2003) indicated that one of the two key principles for Toyota's success was the huge respect shown to people and the well-trained employees forming the cornerstone of its production system. As a coin has two sides, the overdependence on experienced employees has also led to the dilemma of TPS production mode propagation. It took the Toyota Corporation several years and great effort to introduce the TPS production mode to its own production plant in North America by directly exporting a great number of experienced Japanese employees (Womack & Jones, 2003), not to mention the transplantation of the TPS mode into other western companies.

It is natural that humans are involved in the production monitoring process as well. For instance, the famous Jidoka quality control method involved supervision of the manufacturing system by means of "intelligent automation with a human touch" (Ohno, 1988). In other words, although gradually accumulated "poka-yokes" and easily-implementable and effective systems like ANDON are capable of improving the efficiency of a mistake-proofing scheme, human involvement still acts as an

indispensable part of the system. In addition, the supervision of production through Jidoka mainly emphasizes the status of a single work-station; little attention is paid to the healthiness of the entire activity, which is the actual managerial concern.

1.3 A new monitoring concept combining CIM and TPS philosophies

Naturally, the question arises of whether it is possible to combine the advantages of both CIM and Jidoka to oversee the whole production, and this question motivated the originality of this research. The idea arose that it would be quite beneficial if some generic features reflecting the situations of a production system could be extracted, and the requirements for this extraction could prove to be not particularly tedious. In this way, the major obstacles, such as poor-portability and high cost in CIM, could be overcome. With the knowledge of how these features relate to the system conditions, system monitoring could be carried out automatically and the drawbacks in the TPS, such as the inconsistency due to human involvement, could be resolved. Figure 1-2 outlines the advantages/disadvantages of CIM and ANDON.

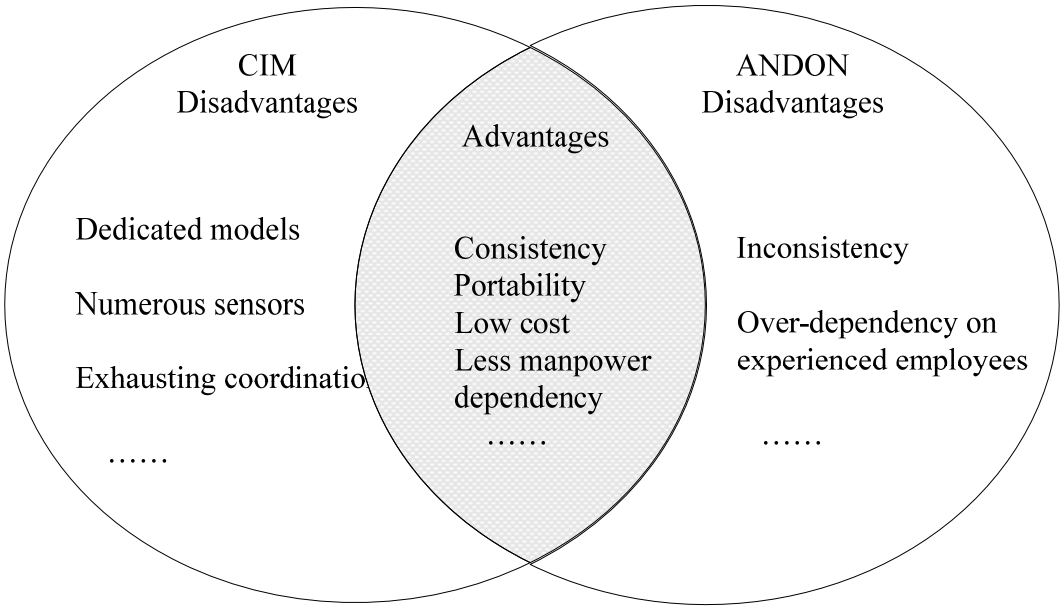


Figure 1-2 Advantages/disadvantages of CIM and ANDON monitoring approaches

To understand a production system, the first step is to create an appropriate representative model. Referring to the International Council on Systems Engineering (INCOSE), a system is defined as a container inside which there are different components interconnecting and working together for the common purpose of meeting a certain requirement (INCOSE, 2004). Apart from the structural components, the interconnections can be in the form of communication, and various types of flows can constitute the characteristics of a dynamic system. For a manufacturing system, it can be a flow model in a high-level sense, with three flows concerned with the normal operation of the entire system: material flow, information flow, and capital flow (Hitomi, 1991). Raw materials flow into the manufacturing system and are converted into finished products by the use of production resources such as machines and labour. Information flow provides channels to help to manage the system effectively, and capital flow plays a crucial role in supporting the expenditure throughout all the stages.

Analogous to the flow model concept, this research abstracts the manufacturing system as a network in which components/work-pieces move through a series of work-stations along defined pathways. Different from the majority of traditional monitoring methods that were built up in a bottom-up manner, it looks at a system from a top-down view in order to reduce the complexity. The basic idea is to reason out the conditions of a manufacturing system based on some features extracted from the flow work-pieces, and doing this can prevent being locked in the trivial operating detail in a system. To a certain extent, this is quite similar to monitoring the circulation system in the human body to figure out the overall healthiness, rather than checking individual organs; this was the primary inspiration for this research, and hence the analogy will be described further in the following section.

1.4 Western and eastern health monitoring philosophies

Generally speaking, based on distinct philosophies, medical diagnosis can be categorized into two different types, the western type and the eastern type. To diagnose a patient, the western approach relies on the information collected from the four key categories of patient interview, physical examination, medical history, and medical tests, while traditional Asian medical practitioners adopt the so-called Four Diagnoses including inspection, listening and smelling, inquiring, and pulse diagnosis. Chang and Yang (2004) compared these two types of diagnostic approaches and found that they are correlated closely with each other (see Figure 1-3). The essential difference between them is in the method of understanding diseases; medical tests in western approaches are commonly conducted through invasive methods, like sampling a piece of a certain organ or an

injection of certain drugs, while eastern approaches rely on non-intrusive techniques such as the pulse diagnosis.

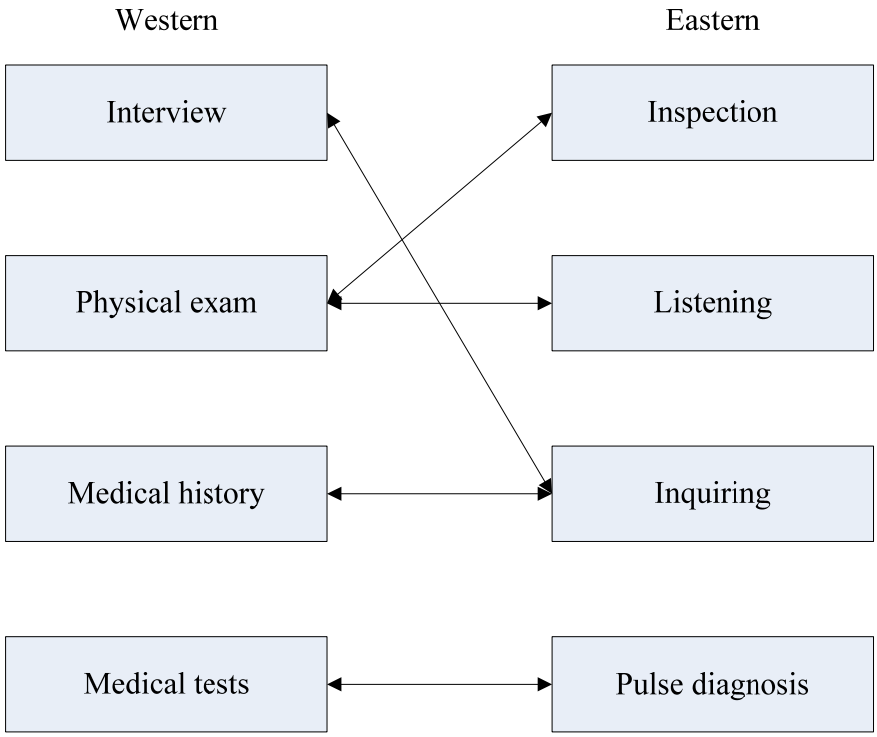


Figure 1-3 Comparison of western & eastern diagnosis methods (Chang & Yang, 2004)

Pulse diagnosis aims at detecting illness from pulse waves measured at six points near the wrist on both hands, called Cun, Guan, and Chi. Pulse waves extracted from these six points represent the health condition and experienced traditional Chinese doctors are capable of distinguishing the characteristics of pulse waves by touching them with their fingertips.

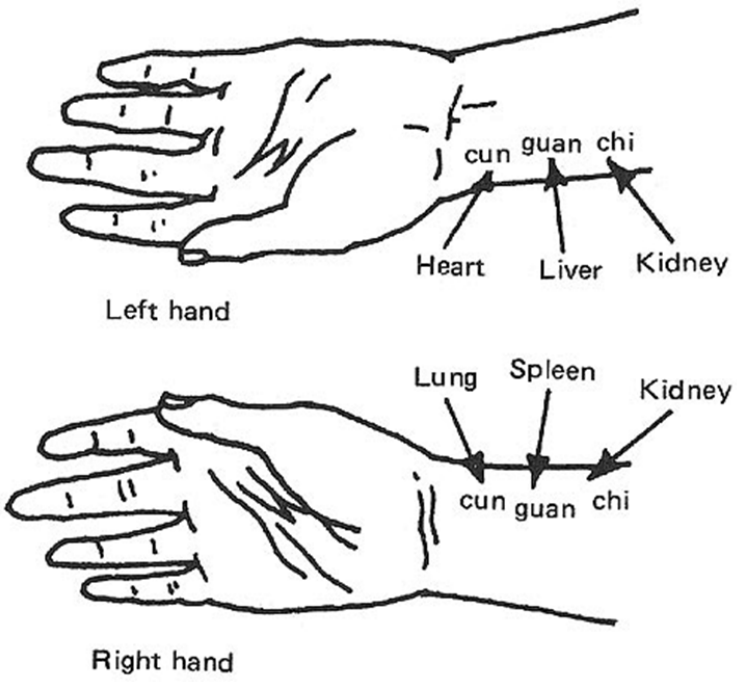


Figure 1-4 Pulse wave measuring positions (Bai, Wu, & Zhang, 1994)

As shown in Figure 1-4, the pulse phenomena measured at the Cun, Guan, and Chi points on the left hand are for checking the health condition of the heart, liver, and kidney respectively, while the same points on the right hand are for the lung, spleen, and the big intestine as well as the kidney. Pulse diagnosis is based on several criteria of the pulse waves, including the variance of the pulse rhythm, shapes of the pulse, pulse depth, pulse interval, pulse smoothness, and pulse strength (Lee et al., 1993). By observing a patient's pulse wave, an experienced doctor is able to identify the disease from which the patient is suffering. Figure 1-5 is the pulse wave signature of a healthy person. Indeed, signatures representing different diseases have already been identified by ancient Chinese doctors and are being complemented gradually.

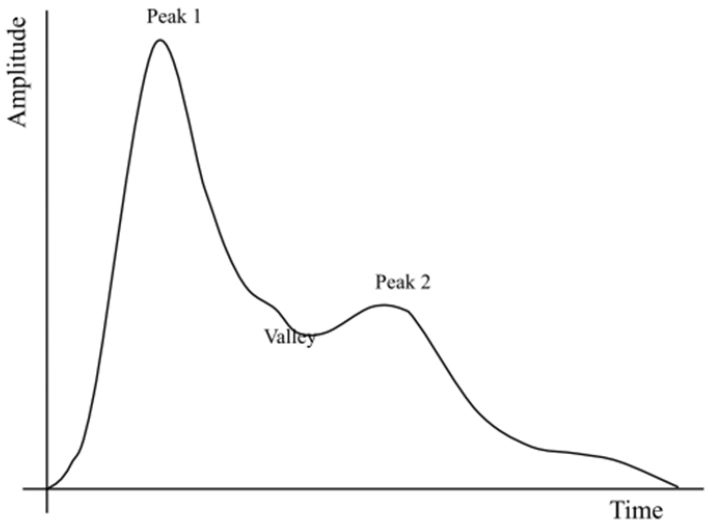


Figure 1-5 Pulse wave signature of healthy person (Bai, Wu, & Zhang, 1994)

1.5 Research objective

Motivated by the inspiration of having the advantages of both CIM and ANDON approaches and illumined by checking the pulse waves to understand the healthiness of a patient, this research set out to explore a new direction in production monitoring; red blood cells are carriers, which are comparable to components moving in a production plant. Similar to the concept of pulse waves indicating illness in a human body, it was anticipated that the components movement would exhibit a different pattern when an abnormality occurred. Therefore, it was anticipated that, by watching their movement patterns, it would be possible to identify various abnormalities.

The main goal of this research was to develop an effective and conveniently-implemented method to monitor the condition of a production system by compensating for the weaknesses of the well-known CIM and ANDON monitoring approaches. Having

this in mind, the sensor devices required can be relatively simple, as we are just interested in knowing how the components are moving there. More specifically, the main objectives to be addressed in this research included:

- To formulate the conceptual framework for representing the dynamic behaviour of the system;
- To extract representative system features and to work out a reasoning scheme to monitor the condition of the system based on the extracted features;
- To develop a method to guide the monitoring system design subjected to specified requirements in practical applications.

1.6 Research contributions

The outcome of this research is expected to make a contribution to system monitoring in the manufacturing field. However, it may also be applicable in other discrete dynamic systems as it is not tailored for any specific industrial system. Actually, the original intention of this research was to generate a generic dynamic monitoring method for a system with which the condition corresponds to the flow entities inside, like manufacturing systems, transportation systems, service systems, or telecommunication systems. The applications of the proposed method are expected to be in the middle management level where one needs to avoid being plunged deeply into the operation particulars.

Chapter Two – Review of literature on manufacturing system monitoring

On the whole, the purpose of this research was to formulate a model to monitor a system based on the dynamic information extracted from the moving entities involved. In the following sections, manufacturing system modelling methods are examined followed by descriptions of the various monitoring approaches. Then, the western and eastern monitoring philosophies are explored. Finally, some fundamental perceptions about the targeted monitoring methodology are outlined to assist the development of the conceptual model in the next chapter.

2.1 Review of manufacturing system modelling

In terms of manufacturing system modelling, generally two types of approaches are mentioned in the literature: System Dynamics Modelling (SDM) and Discrete Event Modelling (DEM). Details about these two types of modelling approaches are discussed below.

2.1.1 System Dynamics Modelling (SDM) approach

When Professor Forrester developed the concept of system dynamics for the first time, the prospective applications were exclusively for business-related and managerial problems in the industrial field (Forrester, 1961). Subsequently, with the essential understanding that feedback and delay govern the behaviour of a system (Richardson & Pugh, 1981), the concept of system dynamics has been applied with a large scope. The fields involved have included public policy analysis (Homer & Clair, 1991), project management (Rodrigues, Dharmaraj, & Rao, 2006), biological study (Hansen & Bie,

1987), software engineering (Abdel-Hamid, 1989), and supply chain management (Angerhofer, 2000). One common characteristic of these systems is that there are information carriers within a system; thus, we can deduce that the moving behaviour of entities with respect to time reflects the real-time status of a system in the sense that they are a kind of information carrier. In fact, how to establish an appropriate model to monitor the network and to regulate the entity flow has become a foundation stone of system dynamics.

In the case of manufacturing systems, SDM has also been studied widely in much academic research. Parnaby (1979) examined the fundamental properties and characteristics of different manufacturing systems and outlined a conceptualized manufacturing system model. Based on the concept of system dynamics, this model is conducive to recognizing the dynamic nature of a manufacturing system clearly. By focusing on three fundamental flows (order flow, material flow and information flow), Edghill and Towill (1989) proposed a generic library of control-theory models to conduct manufacturing system design. They argued that these models would provide great insight into the dynamic behaviour of a manufacturing system. Byrne and Roberts (1994) built an SD model of a Kanban production system by incorporating process flow and information flow. With this model, the performance of the system was easily evaluated, which provided the manufacturer and the supplier with a better understanding of the inter-actions between them. Realizing the lack of applications of SDM in industrial modelling, Baines and Harrison (1999) conducted a survey on the applications of SDM in manufacturing systems. They concluded that system dynamics were more suitable for

aggregated details and predicted that there would be more opportunities for system dynamics in higher level modelling.

2.1.2 Discrete Event Modelling (DEM) approach

Unlike the SDM approach that models a system as a cluster of interconnecting stocks and flows, the Discrete Event Modelling (DEM) approach represents a system as a network of queues and activities with state transitions occurring at discrete time points (Brailsford & Hilton, 2000). Some basic characteristics of a discrete event model include stochastic in nature, ability of handling distinct elements and scheduled events. Owing to the requirement of large amount of operational data, a DEM model is generally suitable for implementation at a tactical level.

One typical DEM approach is the Petri Net that was introduced by Petri in the 1960s. This is a graphical modelling tool suitable for modelling and analyzing discrete event systems. Later, since the 1980s, the applications of the Petri Net have begun to occur in manufacturing systems and some related scope includes modelling, analysis, performance evaluation, scheduling, planning and control (Kahraman & Tuysuz, 2010). Koriem (1999) proposed a R-nets technique to evaluate the performance of hard real-time systems; it was implemented through extending the Time Petri net (TPN) by transforming the probability density functions to uniform distributions over the time intervals associated with the net transitions. Dotoli et al. (2008) showed an on-line fault monitoring technique for discrete event systems based on the first order hybrid Petri Net to address the state explosion problem. Through combining the Coloured Timed Petri Net (CTPN), statistical process control (SPC) and expert systems, Kuo and Huang (2000) presented an integrated model to monitor process failure in Flexible Manufacturing

Systems (FMS). For more information, a comprehensive review of DEM in manufacturing systems can be found in Kahraman & Tuysuz (2010).

To sum up, the major merit of DEM lies in its capability for providing more credible details, while the SDM involves relatively effortless implementation and gives a higher level view. In other words, discrete event modelling is more suitable for operational level modelling. However, when the manufacturing model involves administrative and strategic issues, SDM has a very obvious superiority over the DES (Baines, 1994). For the DEM approach, it is not easy to get a sound picture of the whole system; building such a model for a medium or large-sized manufacturing system is an awesome challenge for engineers due to the elaborate details required, as well as the state expansion problem. In addition, although knowing every detail can be helpful for understanding the system, senior staff may get lost in the trivial routine operation domain and one could be confused when tackling so many issues simultaneously.

To oversee a manufacturing system, it is beneficial to borrow the concept of SDM, taking the system as an integral and examining it in a top-down manner. This also contributes to the original intention of this project, to take a novel look at the manufacturing system from a top-down view and to explore a generic method to monitor the overall process.

2.2 Manufacturing system monitoring methods

New manufacturing and management strategies have emerged continuously in the last several decades, including Computer Integrated Manufacturing (CIM), Lean Manufacturing (LM), Just-In-Time (JIT), and Cellular Manufacturing (CM) (Nagalingam

& Lin, 2008). Monitoring/diagnosis modules act as the essential bridge from specific process problems to some targeted domains by generating notification signals. Almost all processes rely on some form of monitoring techniques to determine operating conditions. Based on the distinct target under monitoring, these techniques can be categorized roughly into two types: Single Machine Oriented Monitoring (SMOM) and Process Oriented Monitoring (POM). In addition, referring to the different reasoning scheme employed, they can also be classified into either direct monitoring or indirect monitoring (Toguyeni & Korbaa, 2005).

2.2.1 Single Machine Oriented Monitoring (SMOM) and Process Oriented Monitoring (POM)

The main purpose of SMOM is to minimize the machine downtime cost so as to enhance the system performance, and research on SMOM covers sensor selection and deployment to root cause diagnosis. The representative SMOM approach is the machine condition monitoring method. Typically, machine condition monitoring has two steps: signal collection and diagnosis inference. Subsequent to the diversification of manufacturing processes, discriminated types of meta-information as well as assorted sensors are applied. Generally speaking, a signal collected can be a piece of direct information, such as the dimensions of a work-piece, or the number of work-pieces passing through a check point, etc.; or one can indirectly monitor the cutting force, vibration, acoustic emission, and so on in order to deduce whether the machine is working well or not. Dimla (2000) offered an extensive review of both, and several commonly-used sensors in machine condition monitoring were reviewed briefly by Dey & Stori (2005). These included cutting force sensors, acoustic emission (AE) sensors, accelerometers and piezoelectric vibration

sensors, current sensors, and motor sensors. Each type of these sensors has its own inherent benefits in applications as well as some drawbacks.

Associated analytical and inferring techniques include neural networks and fuzzy logic theory and numerous literature related to these approaches has been published. Dey and Stori (2005) classified these techniques roughly into four categories: neural networks, fuzzy logic and Hidden Markov models, statistical techniques, and Bayesian networks. In metal cutting process monitoring, Dimla et al. (1997) reviewed the artificial neural network solutions to tool condition monitoring. Sick (2002) also provided an excellent review of the use of neural networks for tool wear monitoring in the turning process, and over one hundred publications can be found in this area. Simulation experiments were conducted and the results of different methods were evaluated. Fuzzy sets and Hidden Markov models, by virtue of their effectiveness in handling obscure boundaries and randomicity, have also been employed in SMOM (Du, Elbestawi, & Li, 1992; Zhu, Wong, & Hong, 2009). Statistical approaches are commonly applied to extract features from time series signals to infer the real-time situation of stations. Bayesian belief networks and diversified evolvements have been suggested, and have been proven by many researchers to be effective in manufacturing monitoring systems (Dey & Stori, 2005; Lewis & Ransing, 1997).

In contrast, POM stresses the overall process condition. Classically, there are three classes of POM in the literature: the quantitative model-based approach, the qualitative knowledge-based approach and the historical data-based approach (Venkatasubramanian, Rengaswamy, & Kavuri, 2003). The model-based approach generates an analytical model to conduct activities and its efficiency depends mainly on

the accuracy of the model. Several review articles of the model-based approach were presented by Kramer and Mah (1993) and Frank et al. (2000). The knowledge-based approach is by means of a qualitative model like a Diagraph or Fault Tree. Historical data-based approaches tend to extract typical characteristics from historical process data, and the monitoring and diagnosis are based on the comparisons of the extracted pattern with the on-line data (Guh, 2010). Representative historical data-based approaches include expert systems, neural networks, etc. The common ground of these monitoring methods is that they either require deep knowledge about the system or depend on a large amount of historical data, which implies extreme effort is required in constructing, extending, and maintaining the system.

2.2.2 Direct and indirect monitoring

Traditionally, a direct monitoring approach employs specialized instruments to collect defined signals from a manufacturing process. Since production processes are often so diversified, discriminated types of meta-information as well as various sensors will be involved. These sensors can either be used to inspect the nature of the work-piece or to collect working information from the tool, like cutting force, noise, thermal and acoustic emission, or vibration. However, this approach is generally only suitable for the monitoring of continuous processes or SMOM (Ly, Toguyeni, and Craye, 2000; Dimla, 2000).

Opposite to a direct monitoring approach, indirect monitoring can also be used to check the status of a system. Ly et al. (2000) presented an approach to monitor the predictive defects in Flexible Manufacturing Systems (FMS) by analyzing the drift rates of the workflows. Having seen the long reactivity of the indirect monitoring approach,

Toguyeni and Korbaa (2005) made improvements to this approach and implemented the cyclic machine scheduling production system. Although the results proved its effectiveness, the shortcoming of rapid resolution trees expansion, along with the increase of system size, remained unresolved. Telmoudi et al. (2008) proposed an indirect technique to detect the deficiency symptoms successfully in FMS. Even so, the inherent character of the indirect monitoring either leads to poor reactivity (Ly, Toguyeni, & Craye, 2000) or is good for a strict scheduling policy only (Toguyeni & Korbaa, 2005). Thus, an indirect monitoring method with both good reactivity and the capability to cope with an unspecified scheduling policy is worth studying by researchers who are working on production monitoring.

In addition, too much monitoring devices intrusion may decrease the consistency and the stability of a system. This is because the compatibility of the various sensor devices can be coarse (Gourgand, Lacomme, & Traore, 2003) and the results are not always satisfactory (Mcgaughey & Roach, 1997). Additionally, a slight system modification in future may become a great challenge. All these inspire the need for careful system monitoring design.

2.3 Manufacturing system monitoring philosophies

Western companies and Japanese companies display two distinct philosophies for monitoring manufacturing system (Qian, Chan, Tang, & Yung, 2011). Normally, western companies employ the Computer-Integrated Manufacturing (CIM) technique to integrate advanced technologies as well as facilities to improve the efficiency. Monitoring modules play essential roles in CIM and each of these modules commonly involves the construction of a dedicated model to address a specific task. To break away from the mire

of complicated model building, Japanese enterprises have combined the modern manufacturing technologies with empirical knowledge from practical practitioners, as in the famous Toyota Production System (TPS) production system.

2.3.1 Computer-Integrated Manufacturing (CIM) philosophy

Many academic research studies concerning dynamic system modelling and monitoring have also been published in the CIM-related literature. Benton and Shin (1998) reviewed the developments of CIM and classified them into two separated types according to the material flow characteristics: Material Requirements Planning (MRP) and Just-In-Time (JIT). The planning processes, monitoring, and control of these two types of CIM systems were compared, as well as their integration methods. Tseng et al. (1999) discussed the design and the implementation of a strategic manufacturing framework based on real time information flow throughout the system. They argued that through this framework, total quality management and MRPII could be achieved. Based on the information flow collected from the distributed inspection points within the system, Piramuthu et al. (2000) gave an adaptive algorithm for dynamic manufacturing system scheduling. Indeed, there are many other CIM system frameworks regarding cutting-edge technologies that have also been proposed (Oztemel & Tekez, 2009; Lewoc et al., 2011; Lin & Jeng, 2006). The covered scope ranges from the operational level to the strategic level, and the techniques involved cover Remote Monitoring, Artificial Intelligence, Distributed Computation, and so on.

Despite the difference in techniques, there are common characteristics among these CIM systems. At the top level of the framework, the data flows interact with good transparency, and to some extent, the over-elaborated involvement of managers can be

avoided. At the operational level, small dedicated models specializing in particular fieldwork are employed. These specific models have to endure procedural dependency constraints or data dependency constraints, or even both (Lara & Nof, 2003), which may bring much inconvenience in later modifications once the original design has been executed for quite a time. A little change to the manufacturing facilities, the processes or some kind of technological innovation can possibly result in the inadaptability of existing models; perhaps, rebuilding the models becomes the sole selection, even though this is both time-consuming and costly.

2.3.2 Toyota Production System (TPS) philosophy

The original goal of TPS is to reduce/eliminate waste in production processes. It is believed that the reliability of a production line and the waste are related closely because the quality defects resulting from facility abnormities will increase waste (Amasaka, 2009). Therefore, production line monitoring has a crucial function in the TPS and this is known as Jidoka. Jidoka is described as "automation with a human touch" (Ohno, 1988) and the main purpose is to identify abnormities rapidly in order to prevent mistakes. Two mechanisms constitute the major components of Jidoka, the Poka-yoke and the ANDON. The Poka-yoke employs simple devices to prevent incorrect operations and can be implemented at any step of a manufacturing process. Whenever an error occurs, ANDON is triggered either automatically by Poka-yoke facilities or manually by operators as an alarm for assistance so that the error can be corrected and further quality problems can be avoided. Although the highly cost-effective nature of the Jidoka monitoring system has won a great reputation, the weakness of overdependence on human intervention impedes the successful implementation of Jidoka outside Toyota and Japan (Liker, 2003).

2.4 Outline of proposed system monitoring method

The CIM approach tends to provide too much detailed information about a system. From the viewpoint of a factory manager, on the one hand he/she wants to provide supervision to a system in time and, on the other hand, to keep away from trivial operational details. In addition, the prerequisite for successful operation monitoring is the task of a specific model, the acquisition and maintenance of which are normally a great challenge for the implementers. On the contrary, Japanese TPS relies little on the model of a particular system but combines Poka-yoka and the human involvement of ANDON to prevent the occurrences of defects. However the application of Poka-yoka also focuses not on the whole system but on a certain operation. The biggest drawback is that too much human involvement has great potential to lead to inconsistency. A novel monitoring method that takes advantage of both CIM and ANDON by using simple facilities with little human intervention in detecting system abnormalities would be helpful.

To monitor a system, the prime requirement is an appropriate model to describe it. Comparable to the traditional Chinese pulse diagnosis method, it is believed that the healthiness of a manufacturing system can be reflected by means of some entity flow patterns inside. By employing simple facilities, information on certain features can be collected at some point in the system, and representative patterns of each feature can also be extracted. This can then be followed by pattern recognition, and abnormality diagnosis. The word “simple” here means not only low cost but also having little variety for different systems. It is anticipated that the system could be modelled uniquely in this way and the proposed monitoring method could be applied directly.

Chapter Three - Formulation of dynamic system monitoring methodology

The ultimate goal of production monitoring is to identify the system conditions so as to provide a control–and-actuation unit with input references for guaranteeing the working smoothness. In the case of CIM, the monitoring process requires a systematic framework to accommodate the signals collected from work-stations/sub-system modules in order to conduct the reasoning operations. The modelling of these work-stations/sub-system modules constitutes the "building bricks" of the monitoring framework and the effectiveness depends strongly on the performance of these bricks. This bottom-up monitoring modelling method is prone to put the production system management in a dilemma. On the one hand, the diversified dedicated "brick" models require extensive input information to guarantee the effectiveness; on the other hand the high-level decision-makers do not expect in-depth technical details, but rather a good picture of the entire system.

Another way to supervise a production system is to have more experienced operators in the monitoring framework, like the ANDON system. Through their rich experiences, it is possible to detect some potential faults through certain early symptoms. However, too much reliance on human interference leads to both inconsistent results and hard generalizations. In fact, the characteristics of discrete parts flowing through the system provide us with valuable clues about the system. In this research, it is anticipated that, by means of building a novel system model, the system level information of a

production floor can be obtained in order for the supervision of the whole plant to become easier.

This chapter starts with a discussion of the conceptual production modelling framework with which the building "bricks" for constructing the entire framework are generated. In conjunction with the proposed framework, the monitoring technique is identified, together with the hardware requirements. Then, the coupled signs representing two general faults (blocking and slowdown) at a production line are explained. Lastly, the chapter addresses how to design the monitoring system to cope with the response time requirement.

To facilitate the presentation, all notations used in the following are listed in Table 3-1. Time series are important factors in this study. To keep a clear distinction between the *clock time* and *time interval*, the capital letter "T" is employed to refer to a time interval while "t" and "*t*" are for a predicted clock time and a measured clock time respectively. For the subscripts in all the notations, the italic characters mean they are variable while the roman ones are an integral part of the linked notation.

Table 3-1 Mathematical notations for the modelling

T_e	Ideal inter-arrival time of components
T_s	Ideal throughput time
$T_{p,k}$	Processing time of the k^{th} work-station, $0 \leq k \leq n$
T_k	Ideal transfer time from the $(k-1)^{th}$ to the k^{th} work-stations, $T_0 = 0$
T_g	Minimum gap between two adjacent components
T_B	Processing time of the bottleneck work-station
$t_{i,\beta}$	Measured clock time when the β^{th} component arrives
$t_{o,\beta}$	Measured clock time when the β^{th} component leaves
$t_{o,\beta}$	Predicted clock time when the β^{th} component leaves
$T_{e,\beta}$	Actual time gap between the β^{th} and the $(\beta-1)^{th}$ components
$\Delta T_{d,\beta}$	Total cumulative time delay of the β^{th} component
T_{BP}	Ideal transfer time from the adjacent upstream work-station to the Blocking Point (BP)

T_{SD}	Altered interval gap between components when serious slowdown occurs
T_i	Time period the abnormality propagating backward to the inlet when blocking/slowdown occurs
T_o	Time period the abnormality propagating forward to the outlet when blocking/slowdown occurs
$T_{rb,max}$	Maximum response time when blocking occurs
$T_{rs,max}$	Maximum response time when slowdown occurs
ΔT_{io}	Time difference between T_i and T_o
K_{BBP}	Index of work-station just before the Balance Blocking Point (BBP) at where $\Delta T_{io} = 0$
T_{BBP}	Ideal transfer time from the adjacent upstream work-station to the BBP
K_{BSP}	Index of work-station just before the Balance Slowdown Point (BSP) at where $\Delta T_{io} = 0$

3.1 System model formulation

The material flow paths in a production system can be considered as connections of branches in each of which components are moving from the inlet to the outlet, from one branch to another. In a branch, a series of work-stations is connected by material transfer mechanisms such as conveyors. From the managerial view, the greatest concern is with the fitness of the whole system rather than that of a certain single work-station or transportation mechanism. When components are flowing in a production branch, they are also carrying information about that branch. Thus, by extracting this information (e.g., the variation of the time span between components, or the cumulative work in progress), the situations in a branch can be determined by particular means. Conceptually, the healthiness of the whole system can be reasoned out by combining details obtained from all the branches. For the modelling purpose, a branch has been named a Region of Interest (ROI) in this research. That is, each ROI serves as a basic module and the ROI model is established first. To collect information on an ROI, the only required devices are a pair of counters (with a time stamped function) installed with one counter at the inlet and the other at the outlet. The reason why we choose this type of device lies in the low

cost as well as the high portability.

Figure 3-1 is a schematic diagram representing a production system, in which several ROIs are presented. In fact, there is no limitation on the number of ROIs in a physical branch and the segmentation of ROIs is subjected to the needs of the system designer. As shown in Figure 3-1, the top-left branch can be either regarded as a single ROI_1 or segmented into $ROI_{1\text{-}1}$ and $ROI_{1\text{-}2}$. In fact, how to segment ROIs in a system to meet the constraint on maximum response time is also an important consideration in this research and is illustrated in a later section. Before the modelling of an ROI, the following hypotheses for the production system should be stated:

(a) Production rate is unchanged at a steady state in normal conditions;

(b) Parts moving in a branch comply with the First-In-First-Out (FIFO) principle;

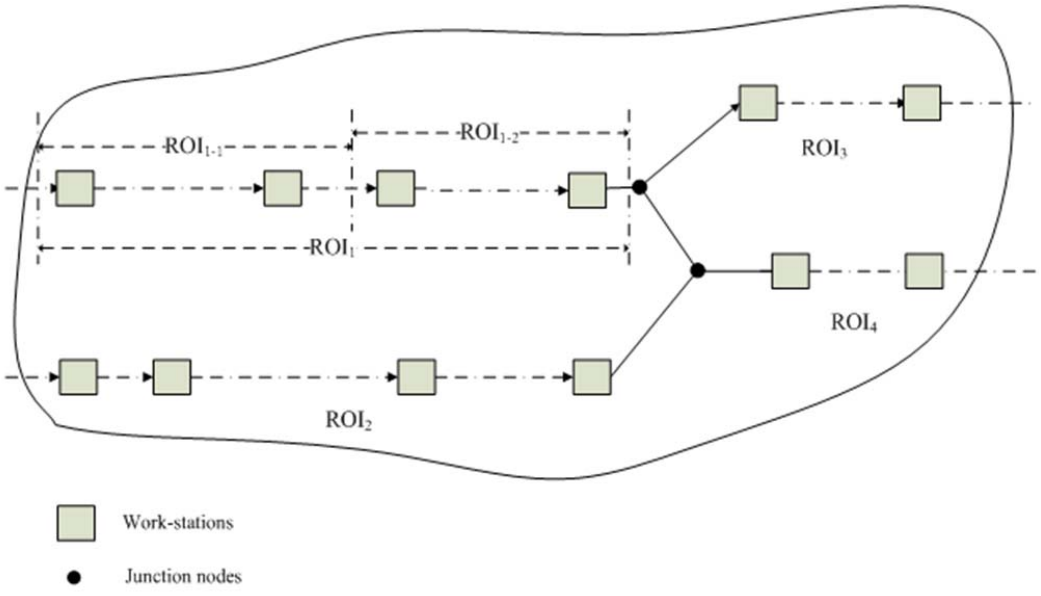


Figure 3-1 Conceptual production network diagram

3.2 Region of Interest (ROI) modelling

Considering a partial production line of several auto work-stations which are connected with material transfer mechanism like the accumulated roller conveyor, in a steady state the processing time of the work-stations are determinate and so is the speed of the transfer mechanism. A component entering the production line is transported by the transfer mechanism from one work-station to another and is processed at all work-stations in sequence. The time duration it spends within the production line can be deduced. This partial production line is defined as an ROI in this research.

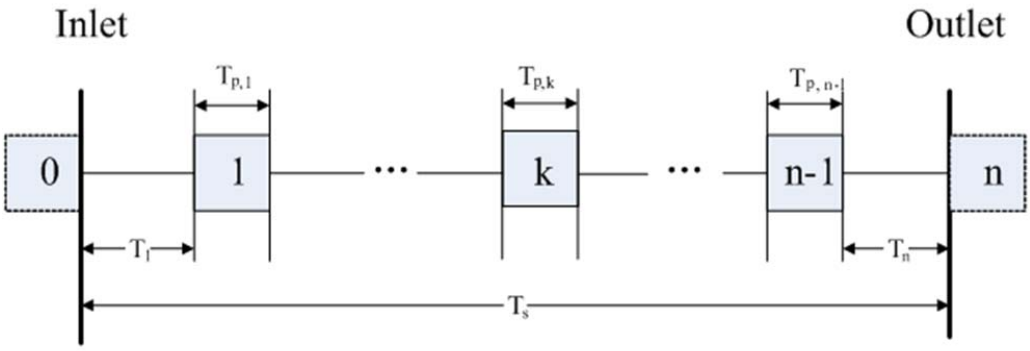


Figure 3-2 Formation of single ROI

Figure 3-2 shows the formation of an ROI; each rectangle represents a work-station (indexed from 1 to n-1) while the two dotted rectangles are the inspection nodes where the counter devices are located. Processing time of the k^{th} work-stations is notated as $T_{p,k}$ while the transfer time from the $(k-1)^{th}$ work-station to the k^{th} work-station is T_k. For the modelling purpose, an inspection node is regarded as a work-station with zero processing time. For a component, the time spent in an ROI is the summation of all the processing time and transfer time.

By recording the clock time when a component enters an ROI and the clock time when it leaves, plus the knowledge of how the component should behave in a normal situation, the circumstances of the ROI can be known. It is obvious that the prerequisite for such knowledge depends on how an ROI is modelled.

3.3 Component moving modelling

When a component enters an ROI in a steady state, its leaving clock time can be predicted. Obviously, if it passes through an ROI without any delay, the leaving clock time is equal to the clock time at which it enters the ROI plus the throughput time of that ROI:

$$\mathrm{t}_{\mathrm{o},\beta} = t_{\mathrm{i},\beta} + \mathrm{T_s}$$

where the throughput time is the summation of all the transfer times and the work-station processing times (also see Figure 3-2):

$$\mathrm{T_s} = \sum_{k=0}^{\mathrm{n}} \mathrm{T}_k + \sum_{k=0}^{\mathrm{n}} \mathrm{T}_{\mathrm{p},k}$$

Like most of the production lines, there is always a bottleneck process and its processing time is represented as:

$$\mathrm{T_B} = \mathrm{MAX}\left(\mathrm{T}_{\mathrm{p},k}\right), \qquad 0 \leq \mathrm{k} \leq \mathrm{n}$$

As one would expect, it is possible for a component to experience some time delay along a production line. In fact, the degree of time delay of a component (say, the β^{th} component) in an ROI is regulated by the time gap with the immediately preceding

component ($(\beta-1)^{\text{th}}$ component) and the time delay suffered by the immediately preceding component. The concept is that the time gap to the immediately preceding component gives a time buffer for a delay to occur. Yet, the time delay of the immediately preceding component may also be influenced by the component just ahead of it, and this is analogous to a kind of induction effect. Taking the β^{th} component for example, the time gap to its immediately preceding component can be obtained as:

$$\mathrm{T}_{\mathrm{e},\beta} = t_{\mathrm{i},\beta} - t_{\mathrm{i},\beta-1}$$

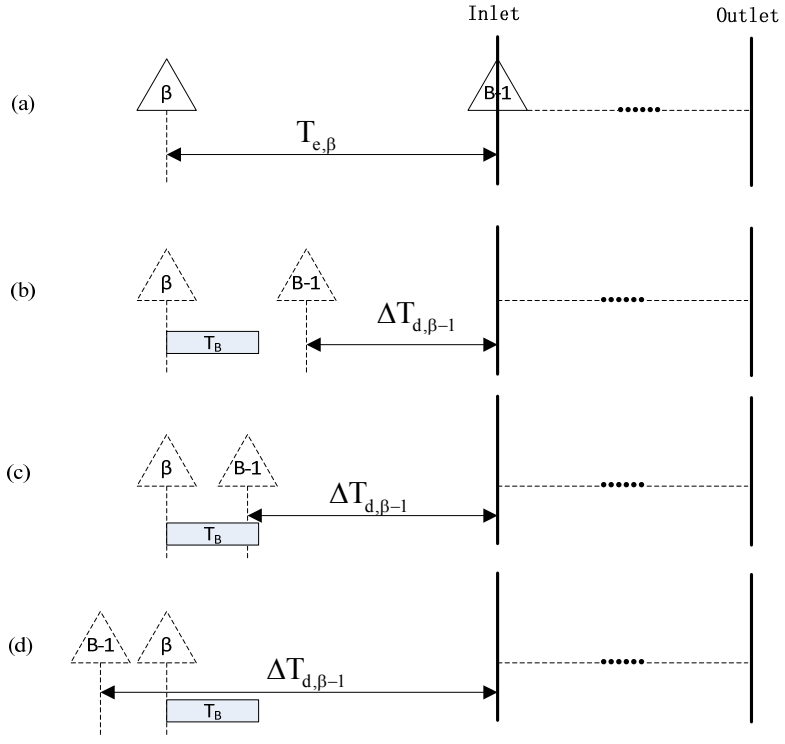


Figure 3-3 Virtual time gap between adjacent components

Figure 3-3(a) shows the case where two components arrive at the Inlet with the time gap of $\mathrm{T}_{\mathrm{e},\beta}$ in an ideal, problem-free condition. To determine the time delay of the β^{th} component, it is necessary to look at the cumulative time delay of its immediately

preceding component ($\Delta T_{d,\beta-1}$). In fact, the influence due to the $\Delta T_{d,\beta-1}$ on the β^{th} component may be absorbed by the time gap between $T_{e,\beta}$ and $\Delta T_{d,\beta-1}$ such that $(T_{e,\beta} - \Delta T_{d,\beta-1})$ is a virtual time gap that can be larger than the processing time of the bottleneck work-station (Figure 3-3(b)), less than the processing time of the bottleneck work-station but still greater than the minimum gap between components (T_g), as in Figure 3-3(c), or less than the minimum gap, as in Figure 3-3(d), where it is possible that the value of the virtual time gap even becomes negative, which means the virtual time gap will not be able to provide any cushioning result at all.

Hence, there are three different scenarios on the β^{th} component. The first is that the β^{th} component encounters no delay if the virtual time gap is greater than the processing time of the bottleneck work-station, as shown in Figure 3-3(b). Second, if the virtual time gap is still greater than the minimum gap (T_g) but smaller than the bottleneck work-station, the delay is the difference between the virtual time gap and the bottleneck processing time (Figure 3-3(c)). Finally, if the virtual time gap is smaller than the minimum gap (Figure 3-3(d)), the β^{th} component closely follows the $(\beta-1)^{th}$ component before the bottleneck work-station but becomes T_B behind the $(\beta-1)^{th}$ component after the bottleneck work-station up to the Outlet. That is, the clock time when the β^{th} component leaves from the ROI is $(t_{o,\beta-1} + T_B)$. Therefore, the time delay of the β^{th} component is the difference between the actual leaving time $(t_{o,\beta-1} + T_B)$ and the ideal leaving time $(t_{i,\beta} + T_s)$. To sum up, the time delay of the β^{th} component should be determined by comparing the virtual time gap with the bottleneck work-station processing time (T_B) and the minimum gap between components (T_g) as:

$$\Delta T_{d,\beta} = \begin{cases} 0, & (T_{e,\beta} - \Delta T_{d,\beta-1}) \geq T_B \\ T_B - (T_{e,\beta} - \Delta T_{d,\beta-1}), & T_g < (T_{e,\beta} - \Delta T_{d,\beta-1}) < T_B \\ t_{o,\beta-1} + T_B - (t_{i,\beta} + T_s), & (T_{e,\beta} - \Delta T_{d,\beta-1}) \leq T_g \end{cases}$$

As a result, the leaving time of the β^{th} component is:

$$t_{o,\beta} = t_{i,\beta} + T_s + \Delta T_{d,\beta} \quad (1)$$

3.4 "Pulse" of ROI

Inspired by the pulse diagnosis method which checks the healthiness of a patient by monitoring the pulses, this research looked into the "pulses" of a production system. Red blood cells are carriers that are comparable to the components moving in an ROI. In principle, by watching the movement of components, it is possible to identify what sort of malfunction, if there is one, is occurring in an ROI. Having this in mind, it is quite natural that a series of "pulses" to reflect the situations of an ROI should first be extracted. It is anticipated that, by combining this "pulse" chains information, the healthiness of the whole system can be visualized. Four distinctive indices that constitute the "pulses" of an ROI are discussed here. To facilitate the description, the β^{th} component is taken as an example in the following.

Regional Inconsistency (RI) - Taking a single component as the target, RI records the difference between the actual measured leaving time and the predicted leaving time of a component to determine system fluctuations. For the β^{th} component, the RI is expressed as:

$$RI_\beta = t_{o,\beta} - t_{o,\beta} \quad (2\text{-}1)$$

Inter-component Arrival Time (IAT) - IAT witnesses the time span between two adjacent components when they enter the ROI successively. In an ideal situation, the components are supposed to arrive at the inlet at a constant interval (T_e). The relationship between IAT and the ROI situation prediction is bilateral. On the one hand, IAT provides the required information ($T_{e,\beta}$) for the prediction of the components' leaving time, which acts as the cornerstone of the proposed monitoring technique. On the other hand, the occurrences of abnormities within an ROI, such as blocking and slowdown, also possibly hinder components from entering the ROI, thereby prolonging the IAT. The IAT of the β^{th} component is presented as:

$$\mathrm{IAT}_\beta = t_{\mathrm{i},\beta} - t_{\mathrm{i},\beta-1} \tag{2-2}$$

Inter-component Leaving Time (ILT) - Similar to the IAT, the ILT logs the interval between two adjacent components when they leave the outlet successively. The deviation of the measured ILT from the predicted ILT provides important clues about an ROI. The definition of the Predicted ILT (PILT), the Measured ILT (MILT), and the deviation of ILT (ΔILT) are as follows:

$$\mathrm{PILT}_\beta = \mathrm{t}_{\mathrm{o},\beta} - \mathrm{t}_{\mathrm{o},\beta-1}$$

$$\mathrm{MILT}_\beta = t_{\mathrm{o},\beta} - t_{\mathrm{o},\beta-1}$$

$$\Delta\mathrm{ILT}_\beta = \mathrm{MILT}_\beta - \mathrm{PILT}_\beta \tag{3}$$

Instant Work in Progress (IWIP) - Instant Work In Progress (IWIP) records the real-time WIP quantity within an ROI. Referring to Little's Law, in a steady state the average WIP within a system remains constant, which is also equal to the multiple of the throughput

time and the production rate. In other words, the fluctuation of IWIP in an ROI can possibly provide valuable hints about what is going on in it.

3.5 Modelling of typical failures in production system

To monitor a production system, it is also quite natural to explore the potential failures to be detected. Typically, failures in a manufacturing system can be classified into two categories, cataleptic and progressive failures (Ly, Toguyeni, & Craye, 2000). Cataleptic failures are sudden and the common consequence is a complete breakdown of either the defective module or even the whole system. Progressive failures refer to the failures that will not discontinue the defective module or stop the system, but reduce the efficiency. To some extent, progressive failures are the portents of cataleptic failures and the timely detection and treatment of a progressive failure can prevent further deterioration of a system. In this project, we examined the situations of blocking (cataleptic failure) and slowdown (progressive failure) in a production line.

3.5.1 Blocking case detection modelling

Blocking stands for stoppage at a certain position in a production branch. Figure 3-4 shows a Blocking Point (BP) located between the m^{th} and $(m+1)^{th}$ work-stations. When blocking occurs, components accumulate upstream of the BP and the overflow propagates to the Inlet after a period of time (T_i). At the same time, there are no more components supplemented to the BP downstream operations and, therefore, after T_o , all components in the BP downstream have left through the Outlet.

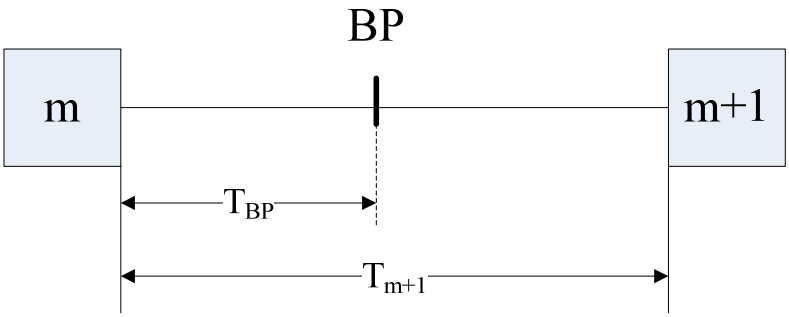


Figure 3-4 Blocking between m^{th} and $(m+1)^{th}$ work-stations

T_i is the time period in which the components queue up to the Inlet, such that the total number of components from the Inlet to the BP increases from n_b to n_e with an inter-arrival time T_e for each component, where n_b is the total number of components before the blocking has occurred:

$$n_b = \frac{1}{T_e}(T_{BP} + \sum_{k=0}^{m} T_k + \sum_{k=0}^{m} T_{p,k})$$

When the overflow propagates upstream to the Inlet, all work-stations are occupied by the components. Supposing each work-station can only hold one component for processing at a time, and there are totally m work-stations from the inlet to the BP, the number of components in between the inlet and the BP is:

$$n_e = m + \frac{T_{BP}}{T_g} + \sum_{k=0}^{m} \frac{T_k}{T_g}$$

Subsequently, the T_i can be calculated by:

$$T_i = T_e * (n_e - n_b) = \frac{(T_e - T_g)}{T_g}(T_{BP} + \sum_{k=0}^{m} T_k) + m * T_e - \sum_{k=0}^{m} T_{p,k} \quad (4)$$

On the downstream side, T_o denotes the duration at which all components lying between the BP and the Outlet have left the ROI and is given by:

$$T_o = \sum_{k=m+1}^{n} \left(T_k + T_{p,k}\right) - T_{BP} \tag{5}$$

Then, the difference between these two durations can be obtained as:

$$\Delta T_{io} = T_i - T_o = T_e * \left[\underbrace{\left(\frac{T_{BP}}{T_g} + \frac{\sum_{k=0}^{m} T_k}{T_g} + m \right)}_{[1]} - \underbrace{\frac{T_s}{T_e}}_{[2]} \right] \tag{6}$$

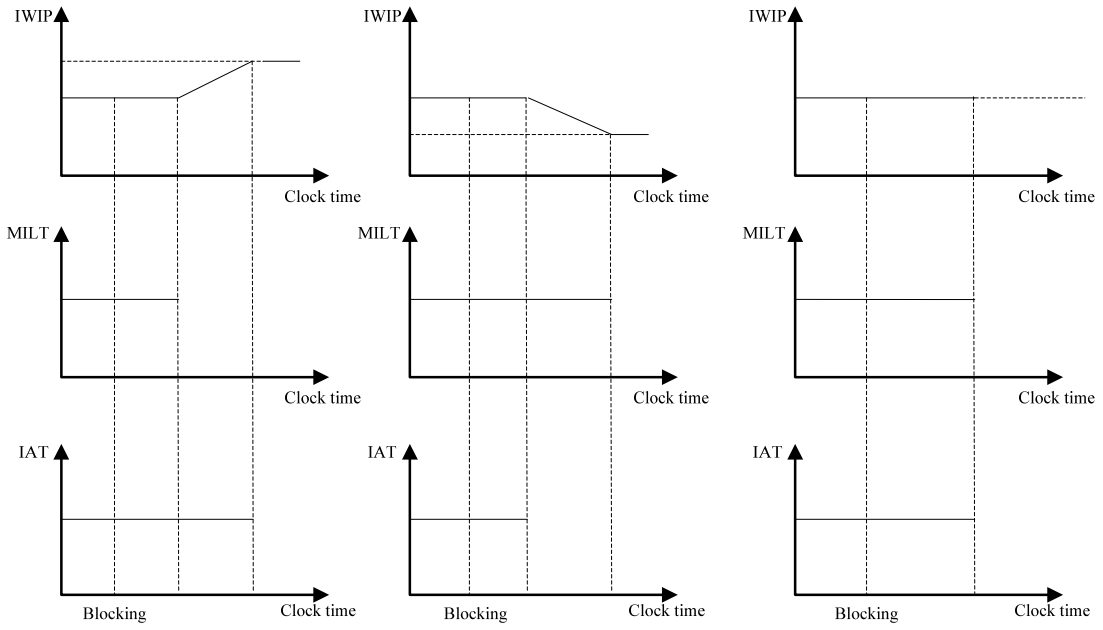
A careful observation shows that sub-formula [1] in Equation (6) is the maximum number of components containable up to the BP while sub-formula [2] is the WIP quantity within the ROI in a steady state. The point at which ΔT_{io} equals zero has a very important role here and it is defined as the Balance Blocking Point (BBP). Given the required parameters of a production system, the location of the BBP (K_{BBP}, T_{BBP}) can be obtained explicitly as:

$$\frac{T_{BBP}}{T_g} + \frac{\sum_{k=0}^{K_{BBP}} T_k}{T_g} + K_{BBP} = \frac{T_s}{T_e}$$

$$\text{s.t.} \begin{cases} T_0 = 0 \\ 0 < T_{BBP} < T_{K_{BBP}+1} \\ 0 \le K_{BBP} \le n - 1 \\ K_{BBP} \in Z^+ \end{cases} \tag{7}$$

In principle, blocking at the BBP can be detected at both the Inlet and the Outlet simultaneously. In other words, for blocking before the BBP ($\Delta T_{io} < 0$), it will be

detected in the first instance at the Inlet and vice versa; it will be signalled at the Outlet earlier than the Inlet if it is after the BBP ($\Delta T_{io} > 0$). Figure 3-5 shows a sketch of the clock time graphs against IWIP, MILT and IAT in the case of Blocking. It is not difficult to see that an increase in IWIP would be observed if ΔT_{io} is a positive value that means blocking occurs after the BBP; the opposite is that a decrease in IWIP would be found which indicates that blocking occurs before the BBP. The calculation of ΔT_{io} can be done by observing IAT and MILT; they are analogous to $(t + T_i)$ and $(t + T_o)$ respectively when they stop respondsing.



(a) Blocking after BBP (b) Blocking before BBP (c) Blocking at BBP

Figure 3-5 Behaviour of IWIP and ΔT_{io} in Blocking

The maximum response time is always an important factor in a monitoring method and it also provides system designers with valuable assistance to deploy a suitable inspection location. In principle, it can be deduced from Equation (6) such that

when ΔT_{io} equals to zero, the blocking occurs at BBP, where the monitoring activity will experience the maximum response time. Thus, Equation (5) becomes:

$$T_{rb,max} = \sum_{k=m+1}^{n} \left(T_k + T_{p,k}\right) - T_{BBP} = T_s - \left[\underbrace{\left(T_{BBP} + \sum_{k=0}^{K_{BBP}} T_k\right)}_{(1)} + \sum_{k=0}^{K_{BBP}} T_{p,k} \right]$$

From Equation (7) the sub-equation (1) can be substituted as follows:

$$T_{BBP} + \sum_{k=0}^{K_{BBP}} T_k = \frac{T_g}{T_e} * T_s - K_{BBP} * T_g$$

Therefore, the maximum response time can finally be presented as:

$$T_{rb,max} = T_s - \left[\frac{T_g}{T_e} * T_s - K_{BBP} * T_g + \sum_{k=0}^{K_{BBP}} T_{p,k} \right]$$

$$= \left(1 - \frac{T_g}{T_e}\right) * T_s + K_{BBP} * T_g - \sum_{k=0}^{K_{BBP}} T_{p,k} \qquad (8)$$

Once blocking has been detected, it will be interesting to position the blocking point in an ROI, and this location can be determined by observing the changes in the IWIP graph. Using the BBP as the datum point, Figure 3-5(a) shows downstream blocking, Figure 3-5(b) is an upstream blocking and Figure 3-5(c) is concerned with blocking exactly at the BBP where the maximum response time will also be obtained. In effect, along with the changing at the IWIP graph with respect to the clock time, the region of blocking can shrink gradually until, ultimately, it is pinpointed. The working principle is to update the IWIP amount continually and, once it is full in the region between the blocking point and the Inlet, the blocking point can be identified if it is

downstream blocking; before that we can only narrow down its region until the T_i has been obtained. Otherwise, if it is upstream blocking, the blocking point can only be located when there are no more components coming out at the Outlet.

In a steady state, WIP has a constant ($\frac{T_s}{T_e}$). In the case where blocking occurs after the BBP, as in Figure 3-6(a), the WIP amount increases by ($\frac{T_i - T_o}{T_e}$) finally. As the potential blocking point further approaches the Outlet, the gap between $(T_i - T_o) = \Delta T_{io}$ also increases, and the time span to the blocking point with reference to the Inlet can be approximated by $\left(\frac{T_i - T_o}{T_e} + \frac{T_s}{T_e}\right) * T_g$; . In case the blocking is before the BBP, the final WIP amount decreases as ΔT_{io} and becomes negative.

In summary, the blocking location is $\left(\frac{\Delta T_{io}}{T_e} + \frac{T_s}{T_e}\right) * T_g$ from the Inlet where ΔT_{io} can be either positive (downstream blocking), negative (upstream block) or zero (at the BBP). In Figures 3-6(a) and 3-6(b), the solid lines show the possible maximum increments and decrements in WIP respectively. In fact, typical downstream blocking will stop somewhere in the positive slope zone as the graph is sketched entirely up to the Inlet, and is similar to an upstream blocking case but with a negative slope.

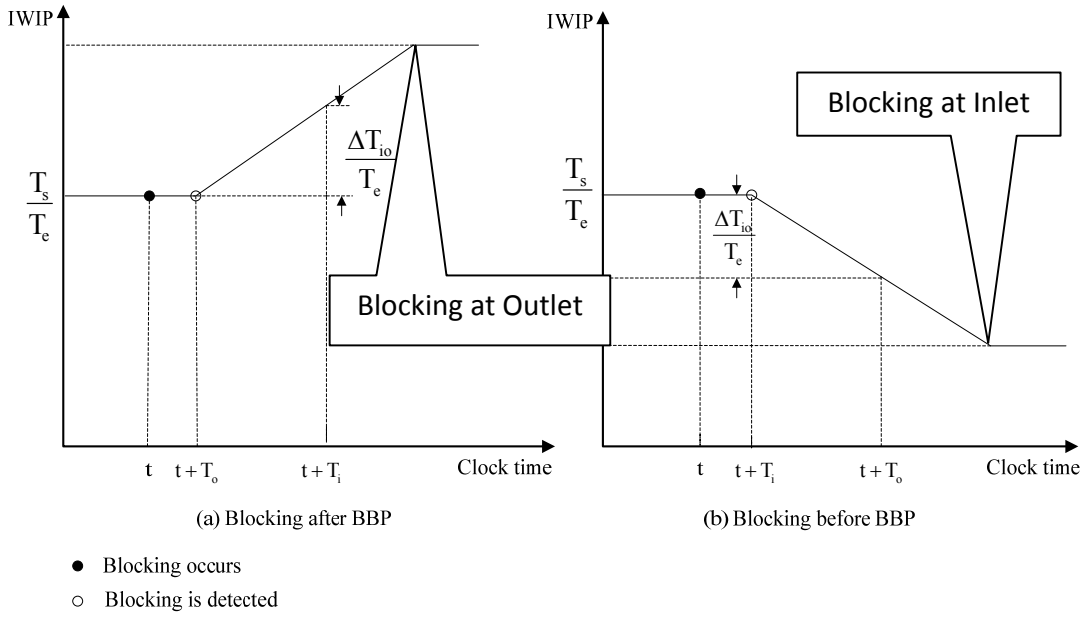


Figure 3-6 Containable WIP against $\Delta\mathbf{T_{io}}$

However, it is of little use only to know the time span from the Inlet to the blocking location, because it is still difficult to locate the actual blocking point, which is not easy to view. To resolve this problem, recall the equation $\left(\frac{\Delta T_{io}}{T_e} + \frac{T_s}{T_e}\right) * T_g$ such that:

$$BP = \left(\frac{\Delta T_{io}}{T_e} + \frac{T_s}{T_e}\right) * T_g = \Delta T_{io}\left(\frac{T_g}{T_e}\right) + \frac{T_s}{T_e} * T_g \quad (9)$$

Obviously, BP is a linear function with slope $(\frac{T_g}{T_e})$ and variable ΔT_{io}, and the constant $\frac{T_s}{T_e} * T_g$ is the y-intercept that is also the BBP value where $\Delta T_{io} = 0$. To cover the entire range of an ROI, one needs to work out the ΔT_{io} at BP = 0 and BP = T_s as in Figure 3-7, and these two points are $(-T_s, 0)$ and $(\left(\frac{T_e}{T_g} - 1\right) * T_s,\ T_s)$. Now, by connecting the two points and inserting the work-station locations in terms of time units along the y-axis,

blocking between any two work-stations can be obtained once the associated ΔT_{io} is known.

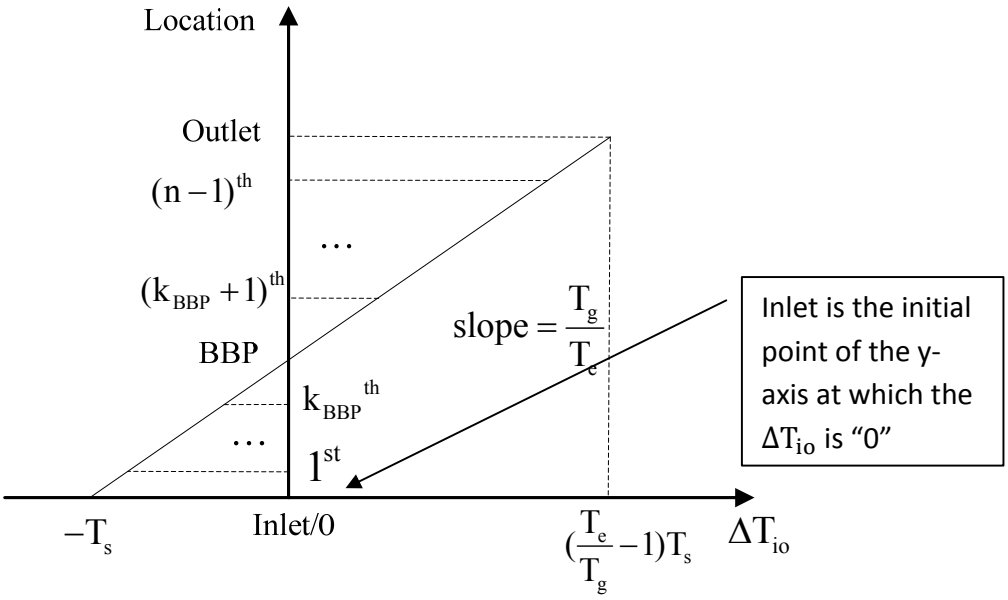


Figure 3-7 Mapping of BP locations onto $\Delta \mathbf{T_{io}}$

3.5.2 Slowdown case detection modelling

In the case of a slowdown in the production line, engineers are concerned about the severity and the location where it happens (see Figure 3-8). Along the production line, the slowdown location can be classified further into either a work-station slowdown or a transfer slowdown. A work-station slowdown involves an efficiency drop; for example, the processing time of the m^{th} work-station is prolonged from $T_{p,m}$ to $T_{p,m}{}'$. Transfer slowdown is concerned with an extension of the transfer time between two adjacent work-stations that may be caused, for example, by a slowdown on the conveyor speed.

Figure 3-9 is a schematic illustration of the transfer time between the m^{th} and the $(m+1)^{th}$ work-station extending from T_{m+1} to $T_{m+1}{}'$. With this information, it can be

deduced that the ILT extends from T_e to $\frac{T_{m+1}'}{T_{m+1}}T_e$. In regard to the severity of a slowdown, it is classified into a slight slowdown or a serious slowdown. In this project, it is defined that the occurrence of a slight slowdown can only lead to an increase of throughput time, while a serious slowdown increases the throughput time and also enlarges the ILT (i.e., system output rate decreases).

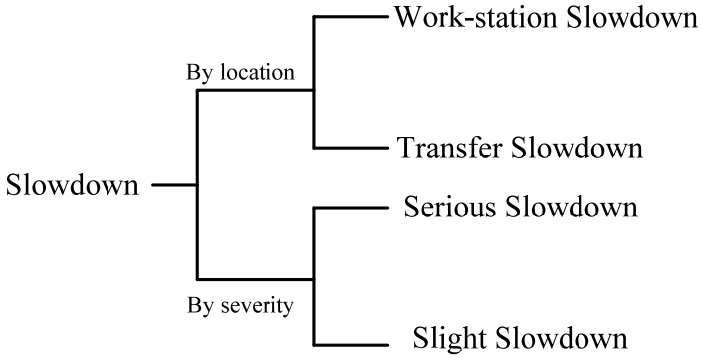


Figure 3-8 Different types of slowdown

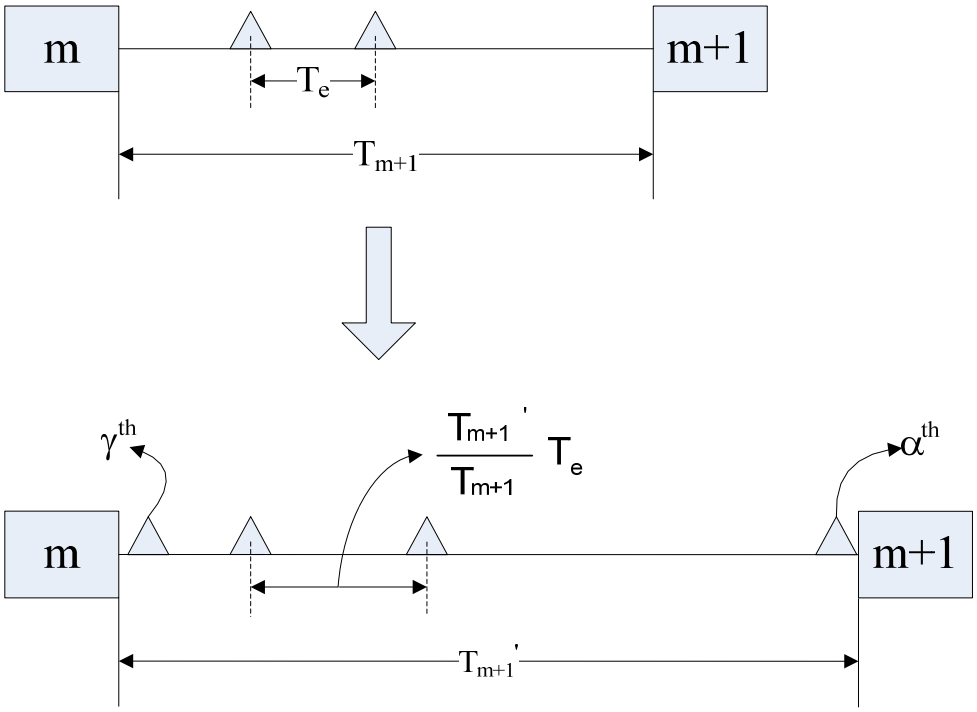


Figure 3-9 Transfer slowdown case

Whenever there is a slowdown, the immediate consequence is an increase in throughput time no matter whether it is serious or not. It would be beneficial for the preliminary screening if one could establish a throughput time threshold to be tolerated (T_t); i.e., a slowdown exceeding the T_t value is classified as a serious event. Indeed, the determination of T_t covers two distinct cases, a serious machine slowdown or a serious transfer slowdown. It is obvious that the root cause of a serious machine slowdown is the increased processing time of a work-station that has exceeded the inter-arrival time (T_e). That means the possible maximum tolerable processing time enlargement for a machine slowdown (T_{tm}) is the gap between the inter-arrival time (T_e) and the minimum processing time among all the involved work-stations (opposite to the bottleneck machine) in an ROI:

$$T_{tm} = T_e - MIN(T_{p,i}) \tag{10}$$

When a transfer slowdown occurs, as in Figure 3-9, the minimum gap between work-pieces increases from T_g to $\frac{T_{m+1}'}{T_{m+1}}T_g$. In the case of ($\frac{T_{m+1}'}{T_{m+1}}T_g$) larger than T_e, the ROI will certainly encounter a queue growing in front of the m^{th} work-station, which will eventually cause an overflow upstream. The output rate of components from the m^{th} work-station decreases and so does the output rate of the ROI; the inter-leaving time of components from the ROI is also enlarged to $\frac{T_{m+1}'}{T_{m+1}}T_g$. On the other hand, if it remains smaller than T_e, the ROI can operate smoothly with the consequent increase in transfer time by an amount ($\frac{T_{m+1}'}{T_{m+1}}T_e - T_e$) only. To conclude, the slowdown can be tolerated if and only if:

$$\frac{T_{m+1}{}'}{T_{m+1}} T_g \leq T_e$$

Thus,

$$T_{m+1}{}' \leq \frac{T_e}{T_g} T_{m+1} \tag{11}$$

And the increase of transfer time can be calculated by:

$$\frac{T_{m+1}{}'}{T_{m+1}} T_e - T_e = \frac{T_{m+1}{}' - T_{m+1}}{T_{m+1}} T_e \tag{12}$$

By substituting the right hand side of Equation (11) into ($T_{m+1}{}'$) at the right hand side of Equation (12), we obtain:

$$\frac{T_{m+1}{}'}{T_{m+1}} T_e - T_e \leq \frac{(\frac{T_e}{T_g} T_{m+1}) - T_{m+1}}{T_{m+1}} T_e$$

$$\frac{T_{m+1}{}'}{T_{m+1}} T_e - T_e \leq (\frac{T_e}{T_g} - 1) T_e$$

This means the maximum tolerance of the transfer time increment for a transfer slowdown (T_{tt}) in an ROI is:

$$T_{tt} = \left(\frac{T_e}{T_g} - 1\right) T_e \tag{13}$$

To consider the effect on both serious machine slowdown and serious transfer slowdown, the maximum tolerance time (T_t) is obtained by combining Equation (10) and

(13) to check which one (T_{tm} or T_{tt}) is larger, because any value exceeding T_t means a serious slowdown has definitely occurred.

$$T_t = MAX(T_{tm}, T_{tt}) = MAX\left\{\left[T_e - MIN(T_{p,i})\right], \left(\frac{T_e}{T_g} - 1\right) T_e\right\} \qquad (14)$$

After consideration on the model formulation, Figure 3-10 shows the **F**our-**L**ayer **F**ilter **A**lgorithm (**FLFA**) that was created to work with the defined features from the "pulses" of an ROI in order to monitor the slowdown symptoms.

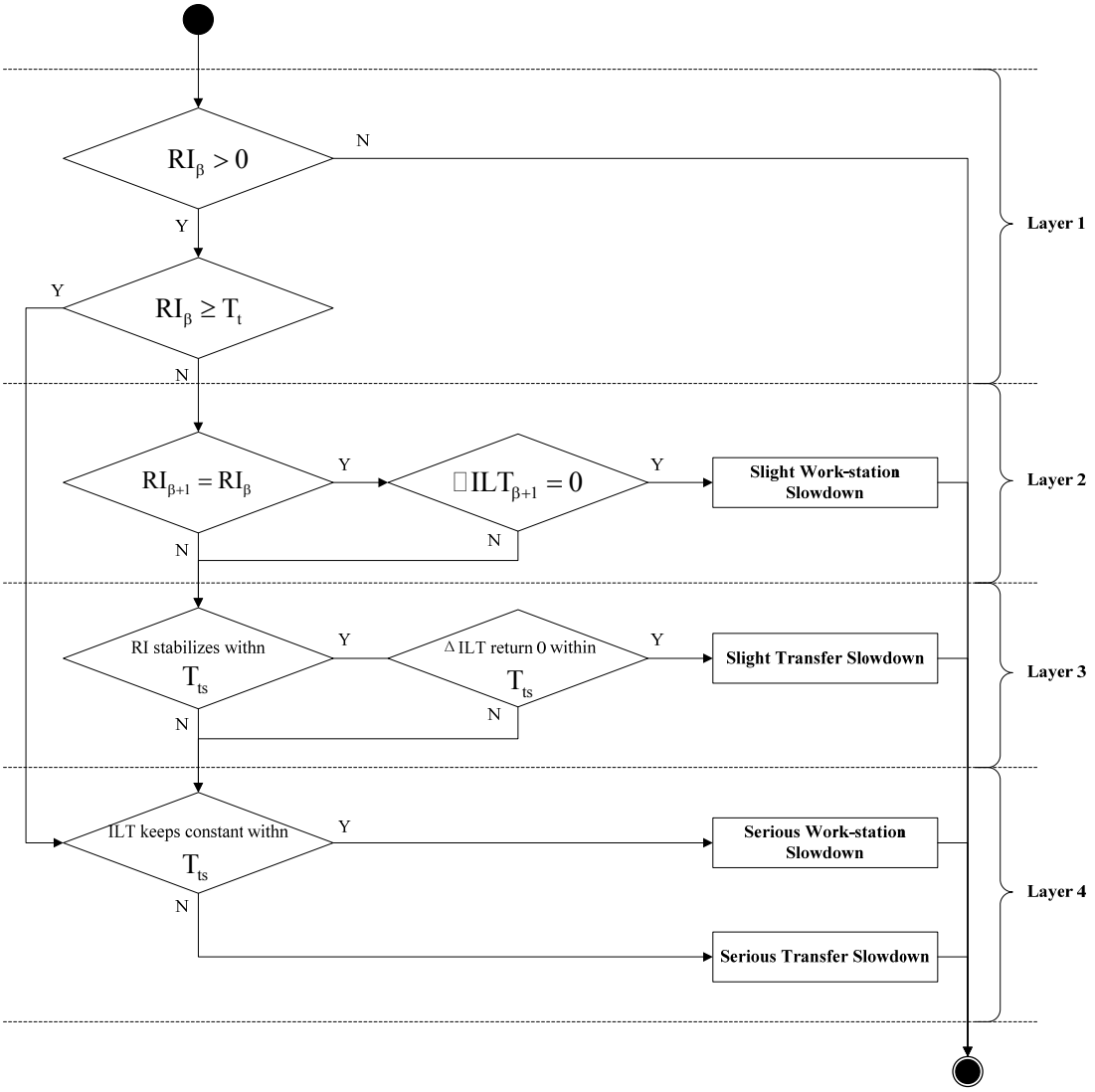


Figure 3-10 Four-Layer Filter Algorithm (FLFA) to monitor slowdown in ROI

Layer 1: The initial observation of a machine slowdown or a transfer slowdown is reflected by the prolonging of throughput time on a component, say the β^{th} component with $\mathrm{RI}_{\beta} = t_{\mathrm{o},\beta} - \mathrm{t}_{\mathrm{o},\beta} \geq 0$. If this RI_{β} is also larger than $\mathrm{T_t}$ (the maximum tolerance time) calculated by Equation (14), it can be deduced that this is a serious one. Then, one can jump to **Layer 4** to judge whether it is a serious machine slowdown or a serious transfer slowdown.

Layer 2: For a slight machine slowdown, components following the β^{th} are anticipated to have the same RI value and the deviations of the measured ILT and predicted ILT (ΔILT) return to 0 after a sudden enlargement. That is to say, by comparing $\mathrm{RI}_{\beta+1}$with RI_{β} and confirming the $\Delta\mathrm{ILT}_{\beta}$ returns to 0, it is a "Slight Machine Slowdown". Otherwise, it is required to go to **Layer 3**.

Layer 3: When a transfer slowdown occurs, supposing the last component on the branch is the γ^{th}one (also see Figure 3-9), the ILT graph will give an obvious alteration after the γ^{th} component flows out of the branch. For a slight transfer slowdown, ILT will change from $\frac{\mathrm{T_{m+1}}'}{\mathrm{T_{m+1}}}\mathrm{T_e}$ to $\mathrm{T_e}$ and the only effect on the ROI is the enlargement of the transfer time from the m^{th} work-station to the $(\mathrm{m}+1)^{\mathrm{th}}$ one. The maximum time duration ($\mathrm{T_{ts}}$) for the ILT kept at $\frac{\mathrm{T_{m+1}}'}{\mathrm{T_{m+1}}}\mathrm{T_e}$ can be obtained as:

$$\mathrm{T_{ts}} = \left[\frac{\mathrm{ILT_{\beta+1}}}{\mathrm{T_e}} * \mathrm{MAX}(\mathrm{T}_k)\right] \tag{15}$$

That is to say, $\mathrm{T_{ts}}$ serves as an upper bound observation time duration for **Layer 3**. Within $\mathrm{T_{ts}}$, if the RI values stabilize at a constant value and ΔILT returns to 0, the

slowdown is classified as a “Slight Transfer Slowdown”. Otherwise, it needs to go further to **Layer 4**.

Layer 4: This layer distinguishes whether it is a serious machine slowdown or a serious transfer slowdown after **Layer 3**. When a serious machine slowdown occurs, the ILT is constant but if there is a serious transfer slowdown, the ILT decreases from $\frac{T_{m+1}'}{T_{m+1}} T_e$ to $\frac{T_{m+1}'}{T_{m+1}} T_g$ within the T_{ts} time duration.

It is anticipated that through these four filtering layers, a slowdown within an ROI can be correctly classified.

3.6 Response time on fault detection

One of the most essential criteria in evaluating a monitoring technique is the maximum response time, which is normally defined as the time duration between a fault occurring to the time that a fault is detected. Regarding the difference in response time, the monitoring and diagnosis tasks for a given manufacturing system can be classified into a immediate response, intermediate response, and slow response (Lee 1998). In fact, there is always a major concern about the maximum response time in designing a monitoring technique and it is quite natural that one should first examine the maximum response time for different cases.

3.6.1 Response time on blocking

With given parameters of an ROI, the location of the BBP (K_{BBP}, T_{BBP}) can be obtained explicitly through Equation (7).

$$T_s = \frac{T_e}{T_g} * T_{BBP} + T_e * K_{BBP} + \frac{T_e}{T_g} * \sum_{k=0}^{K_{BBP}} T_k$$

$$\text{s.t.} \begin{cases} T_0 = 0 \\ 0 < T_{BBP} < T_{(K_{BBP}+1)} \\ 0 \leq K_{BBP} \leq n-1 \\ K_{BBP} \in Z^+ \end{cases} \tag{16}$$

Referring to Equation (16), it is found that once the K_{BBP} (the index of the work-station just before the BBP) is confirmed, there is a linear relationship with the slope of $\frac{T_e}{T_g}$ between T_s and T_{BBP}. In addition, for a confirmed K_{BBP}, the value of T_{BBP} lies between the time interval from the $(K_{BBP})^{th}$ work-station to the $(K_{BBP}+1)^{th}$ work-station. That is to say, to partition ROIs along a given production line, there is a piecewise linear relationship between the length of the ROI (T_s) and the T_{BBP}. In each separate piece, the value of K_{BBP} is constant; also see Figure 3-11 for the sketch of this piecewise linear relationship. For each K_{BBP}, the associated function piece jumps with a new beginning point ($T_{BBP} = 0$) that can be determined by Equation (17).

$$T_s = T_e * K_{BBP} + \frac{T_e}{T_g} * \sum_{k=0}^{K_{BBP}} T_k \tag{17}$$

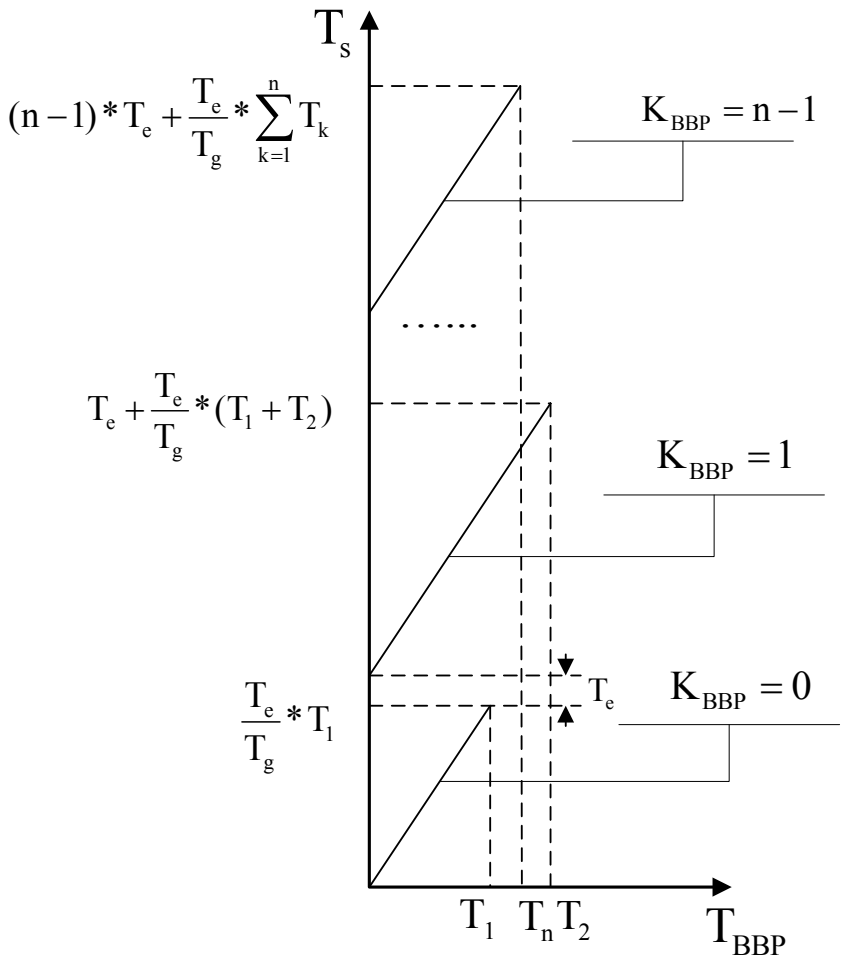


Figure 3-11 Piecewise linear relationship between T_s and T_{BBP}

In a steady state, it is anticipated that the inter-arrival time (T_e) is longer than the required processing time on every work-station. Consequently, a work-station can be regarded as part of the transfer mechanism because each is comparable to a delay in transfer time. Subsequently, the ROI can be simplified as a timeline in Figure 3-12.

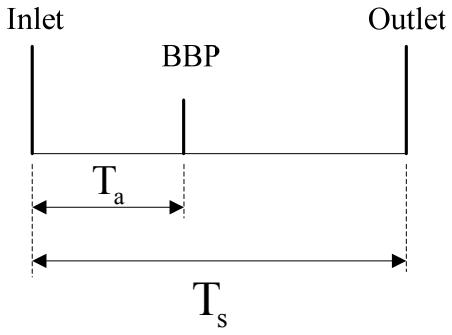


Figure 3-12 Simplified ROI model

Referring to the establishment of BBP, this is at $T_o = T_i$, where T_o is the journey time of a part from the BBP to the Outlet and T_i is the time duration when the parts queue reaches the Inlet. The parts residing between the Inlet and the BBP increase from $\frac{T_a}{T_e}$ in a steady state to the containable upper limit ($\frac{T_a}{T_g}$), as shown in Equations (18-1) and (18-2). The value of T_a is as in Equation (18-3):

$$T_o = T_s - T_a \qquad (18-1)$$

$$T_i = \left(\frac{T_a}{T_g} - \frac{T_a}{T_e}\right) * T_e \qquad (18-2)$$

$$T_a = \frac{T_g}{T_e} * T_s \qquad (18-3)$$

By substituting Equations (18-1) and (18-2) into $T_o = T_i$, the maximum response time for the simplified ROI model ($T'_{rb,max}$) can be deduced.

$$T'_{rb,max} = \left(1 - \frac{T_g}{T_e}\right) * T_s \qquad (18-4)$$

To consider the effect due to work-stations on the maximum response time, we assume that each work-station keeps one part for processing (i.e., no parallel work-stations). Hence, the maximum containable parts between the Inlet and the BBP should exclude the work-stations processing times involved but with the addition of K_{BBP} parts as each work-station contains one part. Thus, the value should be modified from $\frac{T_a}{T_g}$ to $(\frac{T_a}{T_g} - \frac{\sum_{k=0}^{K_{BBP}} T_{p,k}}{T_g} + K_{BBP})$. By substituting this new value into Equation (18-2) and with the assistance of Equations (18-3) and (18-4), the maximum response time with the impact from work-station(s) is:

$$T_{rb,max} = \left(1 - \frac{T_g}{T_e}\right) * T_s + K_{BBP} * T_g - \sum_{k=0}^{K_{BBP}} T_{p,k} \quad (19)$$

It can be observed from Equation (19) that on the occasion where the K_{BBP} is fixed, the maximum response time of a segmented ROI is a linear relationship with the length of the ROI (T_s). Figure 3-13(a) shows $T_{rb,max}$ against T_s at $K_{BBP} = 0$. On the other hand the location of BBP migrates along with the increase of the ROI length, conforming to Figure 3-11. Figure 3-13(b) illustrates the alterations along with this migration, in which the three dotted lines (labelled A, B and C) indicate the step changes with different K_{BBP} values. The trend of $T_{rb,max}$ encounters downward parallel translation in comparison with that at $K_{BBP} = 0$ in Figure 3-13(a) and the associated changes in amplitude are determined by $(\sum_{k=0}^{K_{BBP}} T_{p,k} - K_{BBP} * T_g)$, as each work-station holds one working item at a time. Taking the dotted line A in Figure 3-13(b) as an example, $T_s = \frac{T_e}{T_g} \cdot T_1$ means that when the T_s of an ROI is less than $(\frac{T_e}{T_g} \cdot T_1)$, the BBP

position always lies between the Inlet and the 1^{st} work-station ($i.e., K_{BBP} = 0$). However, if the T_s of this ROI is larger than $\frac{T_e}{T_g} \cdot T_1$, then the BBP position drifts downstream to between the 1^{st} and the 2^{nd} work-station so that the line representing $T_{rb,max}$ against T_s moves down parallel with the amplitude of $(T_{p,1} - T_g)$. In addition, the segmented ROI length (T_s) increases from $\frac{T_e}{T_g} \cdot T_1$ to $T_e + \frac{T_e}{T_g} \cdot (T_1 + T_2)$, and the position of BBP also encounters a shifting downstream to between the 2^{nd} and the 3^{rd} work-stations with amplitude of $(T_{p,1} + T_{p,2} - 2T_g)$. To generalize this situation, if the BBP exists between the m^{th} and the $(m+1)^{th}$ work-stations, the plot of $T_{rb,max}$ against T_s line plotted will be moved down by $(\sum_{i=0}^{m} T_{p,i} - mT_g)$ with respect to the case of $K_{BBP} = 0$.

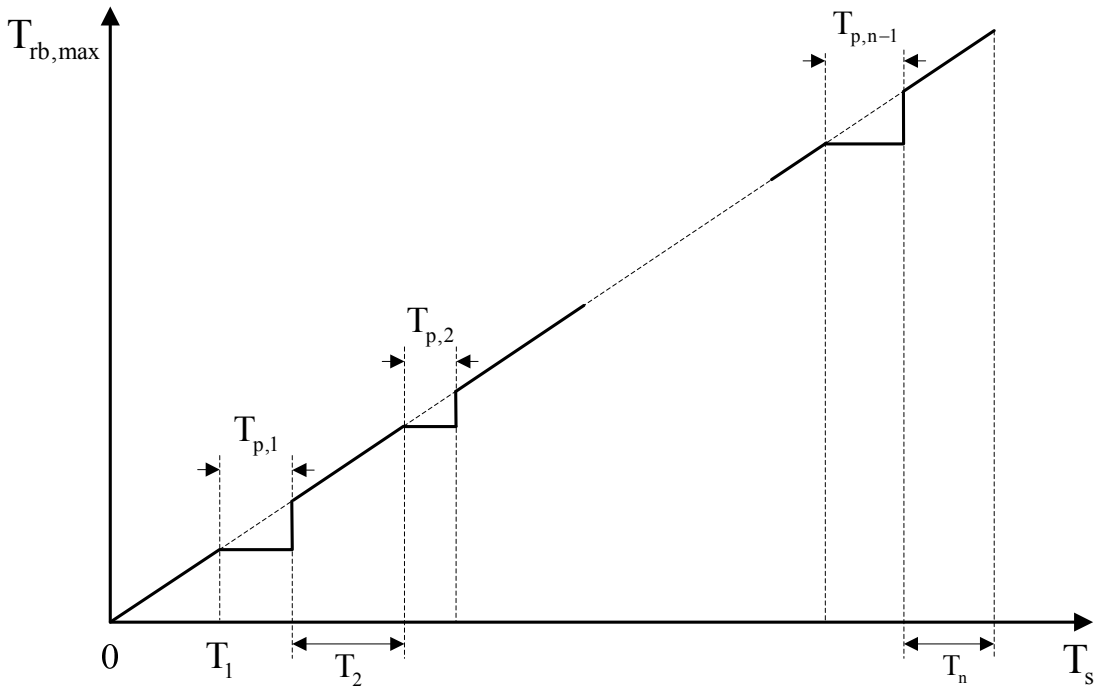
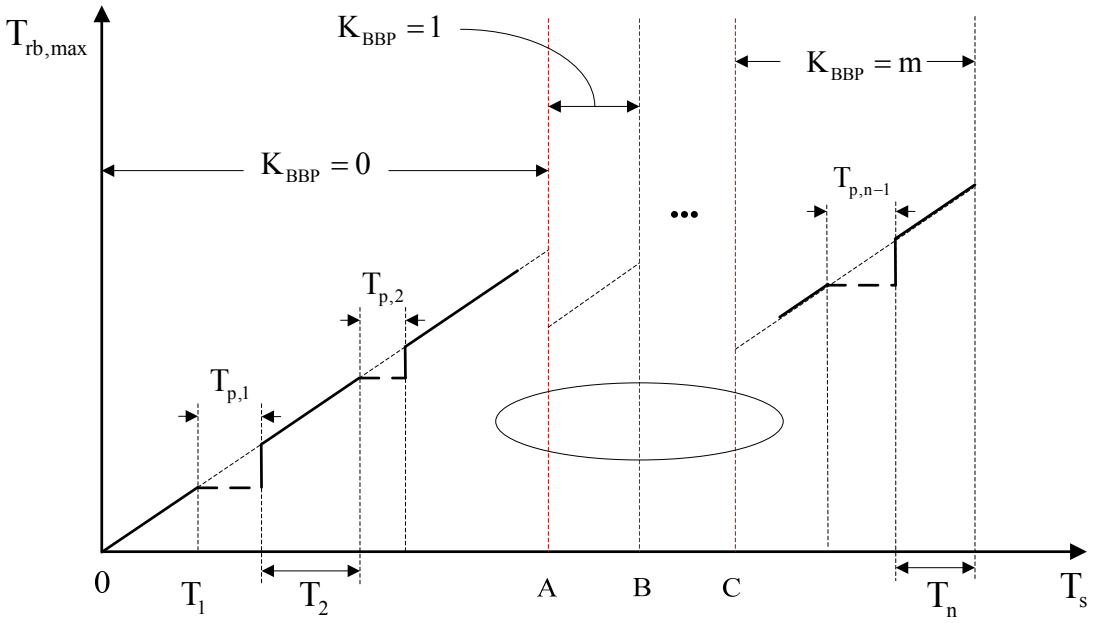


Figure 3-13(a) Relationship between $\mathbf{T_{rb,max}}$ and $\mathbf{T_s}$ when $\mathbf{K_{BBP} = 0}$



$$A: T_s = \frac{T_e}{T_g} * T_1 \qquad B: T_s = T_e + \frac{T_e}{T_g} * (T_1 + T_2) \qquad C: T_s = mT_e + \frac{T_e}{T_g} * \sum_{i=0}^{m+1} T_i$$

Figure 3-13(b) Relationship between $T_{r,b,max}$ and T_s with different K_{BBP}

3.6.2 Response time on slowdown

The response time on slowdown can be analyzed separately based on the varying severity of the slowdown. For a slight slowdown, the only consequence is the enlargement of the component transfer time which will be detected at the outlet of the ROI. That is to say, the response time is equal to the transfer time between the Slowdown Point (SP) and the outlet plus the normal interval gap between adjacent parts. Obviously the slight slowdown will encounter the maximum response time when it occurs at the 1^{st} branch and the maximum response time equals $(T_s - T_1 + T_e)$ time units.

For a serious slowdown, however, the situation is quite different. There is a decrease in the part passing rate at the Slowdown Point (SP) and this can be either at a

sub-branch or a work-station. The result is that parts will gradually congregate between the Inlet and the SP. In addition, the parts moving towards the SP will show some abnormality, and this can be found after all the parts in the downstream of the SP just before the Slowdown occurs have passed the Outlet. Similar to blocking, it is assumed here that it takes T_i for the upstream parts to queue up to the Inlet and T_o for all normal downstream parts to leave from the Outlet; therefore, the response time equals the smaller one of these two values. The point at which there are equal values appearing on both T_i and T_o is called the Balanced Slowdown Point (BSP) and a Slowdown at the BSP will experience the maximum response time.

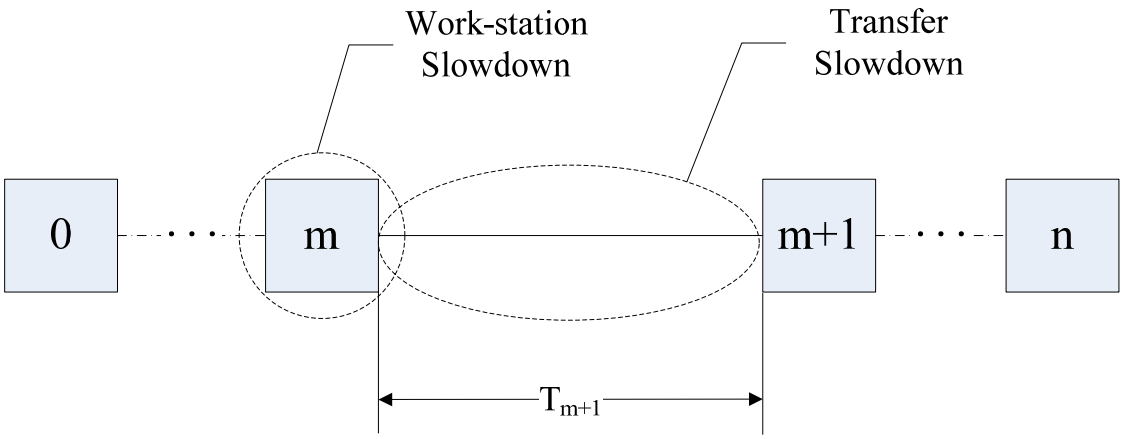


Figure 3-14 Serious slowdown cases in ROI

Supposing a serious slowdown occurs at the m^{th} work-station (a Work-station Slowdown) or between the m^{th} and the $(m+1)^{th}$ work-station (a Transfer Slowdown), as in Figure 3-14. Within T_i time units, the number of components between the Inlet and the SP grows from n_b to n_e, where:

$$n_b = \frac{1}{T_e}\left(\sum_{k=0}^{m} T_k + \sum_{k=0}^{m} T_{p,k}\right)$$

$$n_e = m + \sum_{k=0}^{m} \frac{T_k}{T_g}$$

Supposing that the interval gap between adjacent components passing through the SP is enlarged to T_{SD}, that is to say, the inflow rate to the Inlet is $\frac{1}{T_e}$ while the outflow rate from the SP decreases to the $\frac{1}{T_{SD}}$, then T_i can be calculated as:

$$T_i = \frac{n_e - n_b}{(\frac{1}{T_e} - \frac{1}{T_{SD}})} \qquad (20-1)$$

On the downstream side, T_o is equal to the duration at which all original components lying between the SP and the Outlet have left the Outlet:

$$T_o = \sum_{k=m+1}^{n} (T_k + T_{p,k}) \qquad (20-2)$$

By letting $T_i = T_o$ the location of BSP (K_{BSP}) can be obtained implicitly through Equations (20-1) and (20-2):

$$\frac{T_s}{T_e} = K_{BSP} + \frac{1}{T_g} \sum_{k=1}^{K_{BSP}} T_k + \underbrace{\frac{1}{T_{SD}} \sum_{k=K_{BSP}+1}^{n} (T_k + T_{p,k})}_{[1]}$$

$$s.t. \begin{cases} 1 \le K_{BSP} \le n-1 \\ K_{BSP} \in Z^+ \end{cases} \qquad (21)$$

By comparing Equation (21) and Equation (7), one can observe that the confirmation of the BSP location is similar to the identification of the BBP location in the blocking case. Compared with Equation (7), the T_{BBP} has been eliminated from Equation

(21) as a Slowdown can only occur either on the work-station or on a whole branch ($\mathrm{i.e.}, T_{BBP} = 0$). The newly presented sub-formula [1] in Equation (21) actually indicates the number of components passing through the BSP, from the time the slowdown occurs to the time it is detected at the Outlet. When the gap between components passing through the BSP approaches infinity ($T_{SD} \to +\infty$), Equation (21) becomes the same as Equation (7); approaching a blocking case. In other words, to a certain extent, the blocking case can be regarded as an extremely large slowdown. It is quite natural to consider that there is some relationship between the maximum response time of slowdown and blocking.

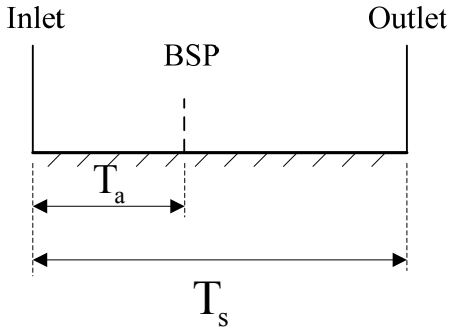


Figure 3-15 BSP location within simplified ROI

Considering the simplified ROI within which the BSP exists at the location of T_a as shown in Figure 3-15. When a slowdown occurs the amount between the Inlet and the BSP increases from $\frac{T_a}{T_e}$ to the containable upper limit ($\frac{T_a}{T_g}$). Hence, according to Equations (20-1) and (20-2), the maximum response time of the slowdown case can be presented as:

$$T_{rs,max} = \frac{(\frac{T_a}{T_g} - \frac{T_a}{T_e})}{(\frac{1}{T_e} - \frac{1}{T_{SD}})} \qquad (22-1)$$

As the maximum response time also equals the transfer time from the BSP to the outlet, the relationship of $T_a = T_s - T_{rs,max}$ exists. By substituting this relationship into Equation (22-1), the maximum response time can be expressed finally as:

$$T_{rs,max} = \frac{(1 - \frac{T_g}{T_e}) \cdot T_s}{(1 - \frac{T_g}{T_{SD}})} \qquad (22-2)$$

It is not hard to find that the numerator of Equation (22-2) is actually the maximum response time of a blocking case (see Equation (18-4)); based on this piece of information, the relationship between the maximum response time of a slowdown and a blocking can be stated finally as:

$$T_{rs,max} = \frac{T_{rb,max}}{(1 - \frac{T_g}{T_{SD}})} \qquad (23)$$

That is to say, for a given ROI, the maximum response time of a serious slowdown is proportional to that of a blocking, and is decided by the severity of the slowdown (T_{SD}).

3.7 Three-Step ROI segmentation technique

For a given production system, to meet the requirement of a specified maximum response time (T_{rc}), the allowable slowdown severity (T_{SD}) is crucial and the most straightforward solution is to segment the production line into separate smaller ROIs as ROI1 and ROI2 shown in Figure 3-1, so that the maximum response time of each one is equal to or less than T_{rc}. Therefore, the designing of the monitoring system is also an ROI segmentation issue. This operation can be carried out systematically by the following three steps.

Step 1: It can be seen from Equation (23) that in an ROI, the maximum response time of a slowdown is always larger than that of a blocking. By taking the specified maximum response time T_{rc} as the tolerable maximum response time of a slowdown and substituting $T_{rs,max}$ in Equation (23), it gives $\overline{T}_{rb,max} = \left(1 - \frac{T_g}{T_{SD}}\right) * T_{rc}$. The modified equation indicates the tolerable maximum response time of blocking in the ROI.

Step 2: Plot the relationship graph of the maximum response time ($T_{rb,max}$) and the length of the production line in time (T_s) by using Equation (19); this is similar to the piecewise linear graph in Figure 3-16.

Step 3: Draw a horizontal line that y-axis equals to ($T_{rb,max} = \overline{T}_{rb,max}$), where the intercept point enables the determination of the associated $\overline{T}_s$ value on the x-axis. $\overline{T}_s$ is the length of the ROI segment that suits the response time requirement. Then one should remove the segment just obtained and repeatedly seek the next segment until the whole production line has been partitioned.

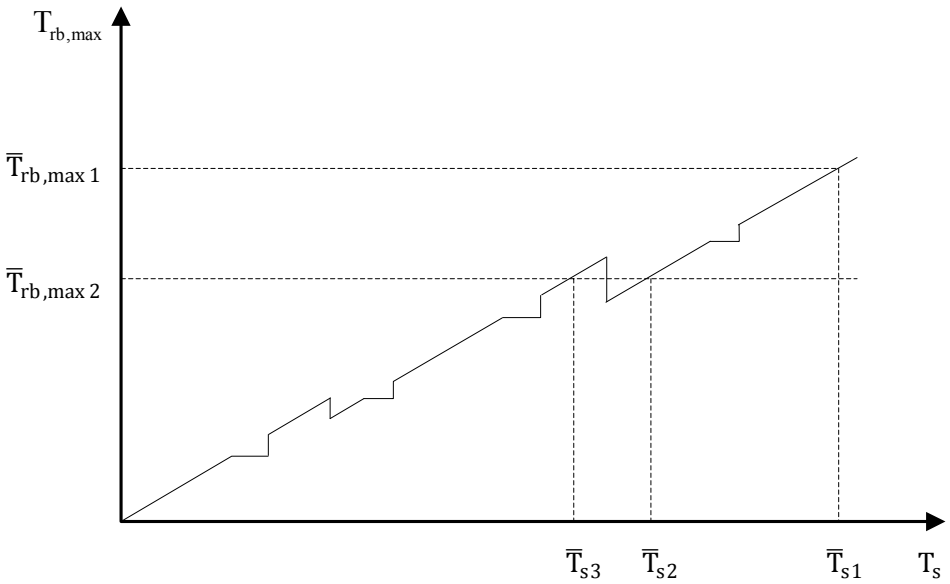


Figure 3-16 ROI segmenting intersections

Figure 3-16 demonstrates some possible issues that may arise, such as when $T_{rb,max} = \overline{T}_{rb,max1}$ there is only one intercept and the length of the ROI segment is confirmed as $\overline{T}_{s1}$. However while $T_{rb,max} = \overline{T}_{rb,max2}$ there are two intercepts ($\overline{T}_{s2}$ and $\overline{T}_{s3}$) with the same maximum response time and normally it would be better to choose T_{s2} as the ROI covers a larger range without additional cost. It is anticipated that based on the iterations of these three steps, one can determine a suitable monitoring arrangement on a production floor so that the failure symptoms can be detected within a specified tolerable time.

Chapter Four - Software implementation for proposed monitoring model

The key element of the proposed method is the prediction of a component leaving time and, based on it, the abnormities within an ROI can be identified. In addition, the BBP location and the maximum response time provide valuable support for the determination of a problematic localization in the application phase and the monitoring system construction respectively in the design phase. In this chapter, the software implementation of these functions is described. C++ is the programming tool used and is compiled into the form of a Dynamic Link Library (DLL) for portability reasons.

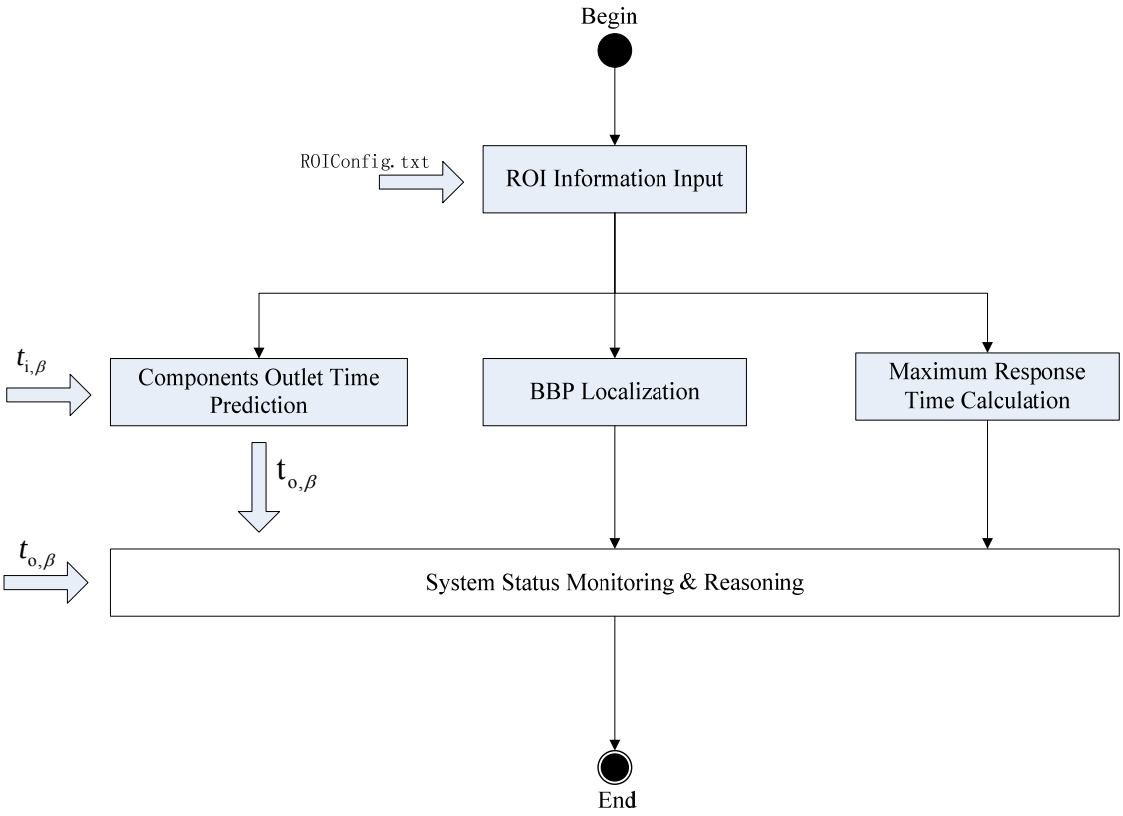


Figure 4-1 Proposed monitoring software framework

Figure 4-1 is the software schematic for the proposed method. For a given production line, the basic monitoring unit in this research is an ROI. First, the input parameters in connection to the targeted ROI should be given. With these parameters the location of BBP and the maximum response time can be calculated, and these two pieces of information are put forward to the System Status Monitoring & Reasoning (SSMR) domain to serve for a fault detection and localization, subject to an allowable time. Once the clock time of when a component arrives at the ROI is known, the Component Outlet Time Prediction (COTP) can predict the leaving time of that component; this is important as the predicted leaving time and the measured leaving time are inputs to the SSMR. Then, by the repeatedly working of the COTP and SSMR, the situation of the ROI can be reflected in real time. The four shaded rectangles in Figure 4-1 are domains implemented by C++ and more details are provided in the following sections. The function of the SSMR is implemented by reasoning the ROI "pulse chain" graphs and is illustrated in Chapter 5.

4.1 ROI parameters input module

The ROI parameters are stored in a separate file in text format (named "ROIConfig.txt" in this case) and the program needs these parameters for the initial configuration purpose. It is understandable that the data within the configuration file has to be arranged in a defined format to suit the software. Figure 4-2 gives the data structure of this configuration file. Although only a single ROI is considered at the moment, future development should enable catering for several ROIs simultaneously. Therefore, a possible extension is embedded and the first line specifies the number of ROIs to be configured. Then the parameters of each ROI are listed one after another. For an ROI, the

total number of work-stations (*WS Quantity*) is recorded first, followed by the minimum gap between components (*TG*) in the next row. Then, it comes to the processing times of the work-stations (*WS * PT*) from the Inlet to the Outlet in order. The last row contains transfer times between work-stations, similar to the line above. Once the parameters in the configuration file have been read by the program, they are organized as the data structure in Table 4-1.

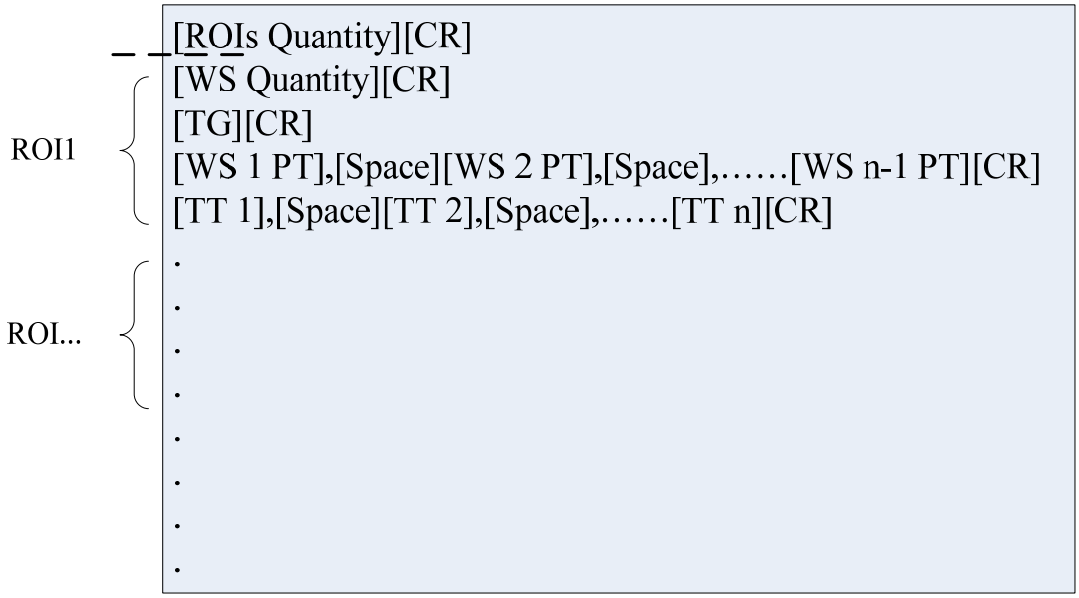


Figure 4-2 Configuration file content arrangement

Table 4-1 Data structure of ROI configuration

```
Struct ROI
{
long int WS_Qty;          //Quantity of Work-stations
long int TG;              //Minimum gap between components
long int* WS_PT;          //Pointer to the work-stations processing times array
long int* TT;             //Pointer to the transfer times array
};
```

4.2 BBP localization

The BBP determination plays an important role in abnormal position pinpointing. There are two parameters: the work-station index just before the BBP (K_{BBP}) and the ideal transfer time from the immediate upstream work-station to the BBP (T_{BBP}). Referring to Equation (7) in Chapter 3, the location of BBP is also related to the ideal inter-component arrival time. Therefore, the BBP localization function includes one input and two output parameters (See Table 4-2).

Table 4-2 BBP identification function arguments

```
int BBP_Identification (long int _iT_e, long int & _iK_bbp, long int & _iT_bbp);
/*!
 *@fn identify location of the BBP
 *@param[in] _iT_e The ideal inter-component arrival time
 *@param[out] _iK_bbp The index of work-station just before the BBP
 *@param[out] _iT_bbp The transfer time from its previous work-station to the BBP
 *@return 0 for success or -1 for fail
*/
```

4.3 Maximum response time calculation

As mentioned before, the maximum response time of an ROI is a critical specification in monitoring system design. It has been spelt out in the formulation of Equation (23) that once the tolerable slowdown rate is confirmed, the maximum response time of the system (also that of a slowdown case) can be expressed easily as the maximum response time of the blocking case. In other words, to meet this system design requirement, only calculating the maximum response time of the blocking case *MRT_Cal()* is needed. The arguments of *MRT_Cal()* are shown in Table 4-3. The output is the maximum response

time of the blocking case (*_iT_mrt*) while the input is the ideal inter-component arrival time (*_iT_e*). The BBP position is also needed in the implementation and can be obtained through the function *BBP_Identification()*.

Table 4-3 Function interface of maximum response time calculation

```
int MRT_Cal (int _iT_e, int & _iT_mrt);
/*!
 *@fn Calculate maximum response time
 *@param[in] _iT_e The ideal inter-component arrival time
 *@param[out] _iT_mrt The maximum response time of blocking case
 *@return 0 for success or -1 for fail
*/
```

4.4 Components outlet time prediction

Information on an incoming component is important in the proposed monitoring method and the function *New_Comp()* is for the creation of new component data. When a component arrives at an ROI Inlet, the clock time is first recorded as the arrival time is the sole input parameter for the component. With the arrival time and the known configuration parameters of the ROI, its departure time can be predicted; see $t_{o,\beta}$ on page **27** and Equation (1) on page **34**. The determination of the time delay within the ROI is relatively complicated and hence, the *Get_Comp_Delay()* function has been coded to help to obtain the predicted leaving time function *Get_Comp_Outlet_Time()*. In addition, several transition functions have been constructed to assist the prediction including *Set_Comp_Arrival_Gap(), Get_Comp_Arrival_Gap()* and *Get_Comp_Delay()*. The descriptions of these functions can be found in Table 4-4 and details can be referred to in Appendix F.

Table 4-4 Component leaving time prediction functions

```
long int New_Comp(long int _it_i);
/*!
 *@fn Create new component for the program, index of the component is generated
 *@param[in] _it_i The clock time when the component arrives at the inlet
 *@return 0 for success or -1 for fail
 */
```

```
long int Get_Comp_Outlet_Time (long int _iComp_Index);
/*!
 *@fn Retrive the predicted component outlet time
 *@param[in] _iComp_Index The index of the component
 *@return x for success or -1 for fail       x means the predicted outlet time
 */
```

```
long int Get_Comp_Delay (long int _iComp_Index);
/*!
 *@fn Retrive the total time delay of the component within the ROI
 *@param[in] _iComp_Index The index of the component
 *@ return x for success or -1 for fail       x means the time delay
 */
```

```
long int Set_Comp_Arrival_Gap (long int _iComp_Index);
/*!
 *@fn Calculate the inter-arrival gap between the component and its previous one
 *@param[in] _iComp_Index The index of the component
 *@ return 0 for success or -1 for fail
```

```
 */

---

long int Get_Comp_Arrival_Gap(long int _iComp_Index);
/*!
  *@fn Retrive the inter-arrival gap of the component
  *@param[in] _iComp_Index The index of the component
  *@return x  for success or -1 for fail       x means the inter-arrival gap
  */
```

4.5 ROI segmentation

The three-step ROI segmentation technique has been employed to automate the graph plotting and the ROI segmentation described in Section 3-7 (Also see Table 4-5). With input arguments about the ideal inter-component arrival time (*_iT_e*) and the tolerable maximum response time of the blocking case (*_iT_rc*), the *ROI_Segment()* function provides a convenient way to segment ROIs along a given production line. The output results include the length of the segmented ROI (*_iTs_Target*), the index number of the immediate upstream work-station (*_ik_ws*), and the length in time duration between the work-station and the segmentation point (*_iT_position*). Once an ROI is segmented from the production line, the interface *ROI_Reset()* should be invoked to remove the just obtained ROI so that the further segmentation can proceed. Normally by combining the two interfaces of *ROI_Segment()* and *ROI_Reset(),* a production line can be segmented continuously into ROIs until the *ROI_Segment()* returns "Fail". The last un-segmented part is the final ROI that can be obtained by calling the *Get_Final_ROI()* function.

There are cases where the segmentation points are not to be positioned on a transfer branch so that the time-stamp counter can only be installed at a work-station.

This means that the segmentation point falling at a work-station and the exit of a work-station has been selected as the potential segmentation point. With this constraint, the function *ROI_Segment_WB()* is implemented. The basic job of the *ROI_Segment_WB()* is the same as the *ROI_Segment()*, except that the parameter *_iT_position* is not in the output parameters list; it means that the segmentation points will always lie at the exit of the work-stations (i.e.,*_iT_position =0*).

Table 4-5 ROI segmentation functions

```
long int ROI_Segment(long int _iT_e, long int _iT_rc, long int& _iTs_Target, long int& _ik_ws, long int& _iT_position);
/*!
*@fn Segment an ROI from the production line
*@param[in] _iT_e The ideal inter-component arrival time
*@param[in] _iT_rc The tolerable maximum response time of blocking case
*@param[out] _iTs_Target The length of the segmented ROI
*@param[out] _ik_ws The largest index of the work-stations in the target ROI
*@param[out] _iT_position The length from the (_ik_ws)th work-station to the segment point
*@return 0 for success or -1 for fail
*/
```

```
long int ROI_Segment_WB(long int _iT_e, long int _iT_rc, long int& _iTs_Target, long int& _ik_ws);
/*!
*@fn Segment an ROI from the production line
*@param[in] _iT_e The ideal inter-component arrival time
*@param[in] _iT_rc The tolerable maximum response time of blocking case
*@param[out] _iTs_Target The length of the segmented ROI
```

```
*@param[out] _ik_ws The largest index of the work-stations in the target ROI
*@return 0 for success or -1 for fail
*/
```

```
long int ROI_Reset(long int _iTs_Target, long int _ik_ws, long int _iT_position);
/*!
*@fn Segment an ROI from the production line
*@param[in] _iTs_Target The length of the segmented ROI
*@param[in] _ik_ws The largest index of the work-stations in the target ROI
*@param[in] _iT_position The length from the (_ik_ws)th work-station to the segment
point
*@return 0 for success or -1 for fail
*/
```

```
long int Get_Final_ROI(long int& _iTs);
/*!
*@fn Get the last segmented ROI (to the outlet)
*@param[out] _iTs The length of the final ROI
*@return 0 for length>0 or -1 for length ==0
*/
```

4.6 User interface in Excel

The above functions were compiled into a DLL library for ease of accessing by multiple programming languages. In this project, Visual Basic for Application (VBA) language was used with Microsoft Excel to develop the user interfaces and to manipulate the functions. With the aid of Microsoft Excel in data handling and curve generation, the System Status Monitoring & Reasoning module was built to present the system "pulse" graphs. In fact, the experiments described in Chapter 5 made use of this program.

Two user interfaces are founded in the Excel worksheet. Figure 4-3 is the "Component Prediction" user interface for the components leaving time prediction. First of all, the system capacity in terms of the maximum number of work-stations ① and ROI configuration parameters ② should be input manually or read from the configuration file "ROIConfig.txt" by clicking the "Read Parameter" button on the user interface. Then the BBP location of the ROI can be obtained by clicking the "Get BBP" button and the results are shown in the cells range ④. When the actual arrival time of components are imported into area ③, the predicted leaving time will be presented in output area ⑤ by clicking "Run Prediction". The last column records the actual outlet time of components which is used to compare with the predicted ones in area ⑤.

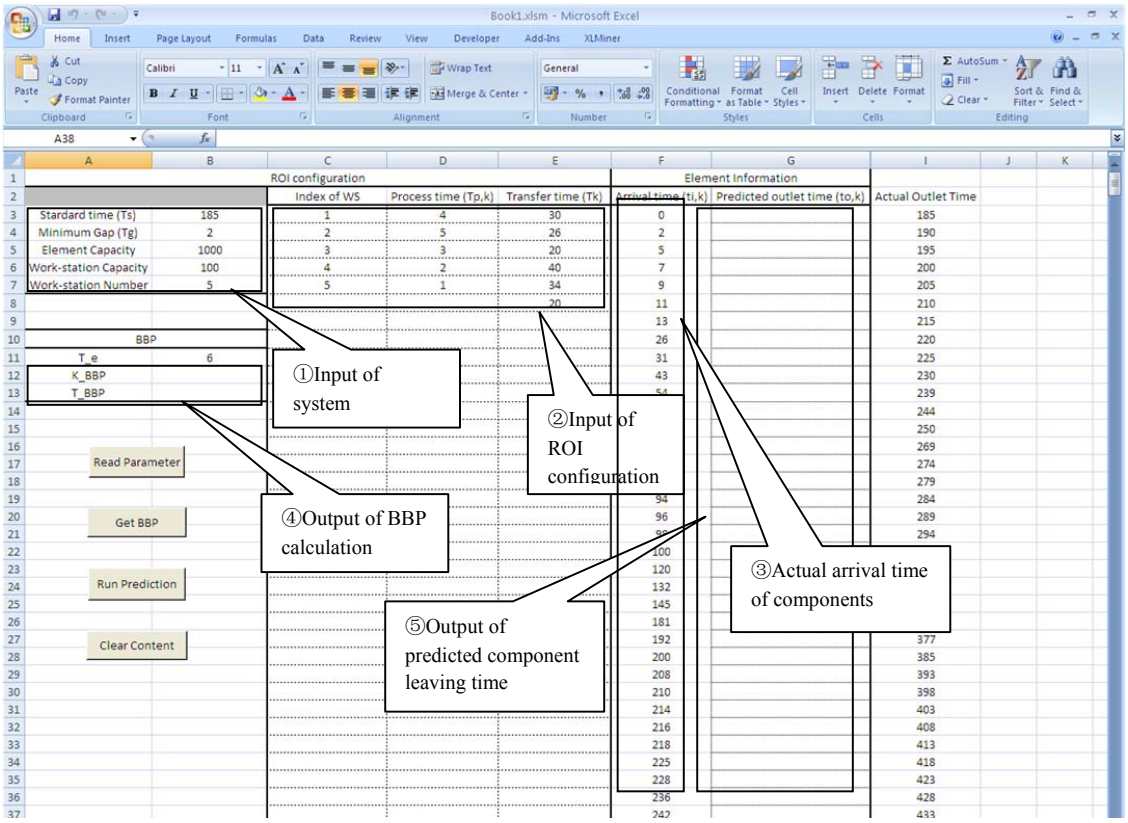


Figure 4-3 User interface for components leaving time prediction

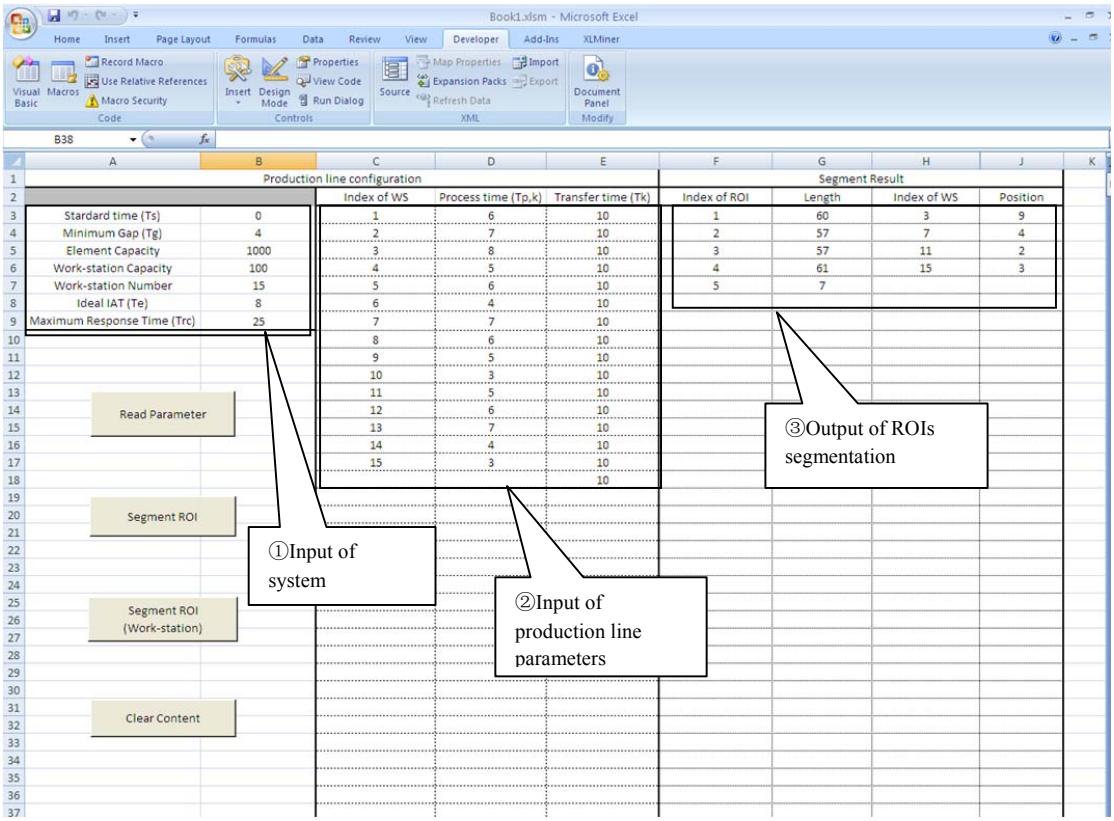


Figure 4-4 ROIs segmentation user interface

Another user interface is the “ROI Segmentation” as shown in Figure 4-4 and it is implemented to perform the ROIs segmentation. Similarly, the system capacity information and production line configuration parameters should first be given to the cells range ① and ② in Figure 4-4 respectively. Then by clicking either the “Segment ROI” button or the “Segment ROI (Work-station)” button, the segmentation results will be shown in area ③. The difference is that the “Segment ROI” button invokes the *ROI_Segment()* function while the “Segment ROI (Work-station)” calls the *ROI_Segment_WB()* function. The “Length” in the output area ③ refers to the length of the time period covering each segmented ROI. The “Index of WS” means the index of work-station just before the segmentation point and the “Position” indicates the time

duration from that work-station to the segmentation point. Additionally, the “Position” column becomes zero in the case of applying the *ROI_Segment_WB()* function. The last segmented ROI only presents the length in area ③ because the segmentation point actually lies at the exit of the production line.

Chapter Five - Case study and analysis

It can be seen from Chapter 4 that the implementation of the functions is rather straightforward, which also complies with the project's original intention of being "Simple & Generic". This being the case, these functions act as the cornerstone of the subsequent monitoring activities and their accuracy needs to be guaranteed first. This chapter starts with the validation of these functions by comparing the output of the interfaces with the results of a hand calculation. After that, there is a description of different defect scenarios within a single ROI that were conducted to test the validation of the proposed monitoring method; a reasoning scheme to diagnose the ROI by means of observation of systematic "pulses" is then summarized. The final section shows the steps of the monitoring framework design in a long production line by means of ROI segmentations subjected to a specified maximum response time.

5.1 Validation of functions through handy simulation

Table 5-1 shows the parameter configuration of an ROI with five work-stations. In an ideal state, components are supposed to arrive at the inlet with constant time intervals of 6 time units. By substituting the parameters given in Table 5-1 into Equation (7) to search for a feasible solution, after trials it is obtained that K_{BBP}=1 and T_x =24 in this case. This means that the BBP is at 24 time units after the 1^{st} work-station. The software output can be obtained conveniently by invoking the function *BBP_Identification()* and the result is the same as the hand calculation result (see Figure 5-1).

1^{st} trial:

input $K_{BBP} = 0$ then $T_{BBP} = 62$;

as there is the constraint that $0 < T_{BBP} < 30(T_1)$,

∴ The result is invalid

2^{nd} trial:

input $K_{BBP} = 1$ then $T_{BBP} = 24$;

according to the constraint that $0 < T_{BBP} < 26(T_2)$,

∴ The result is valid

Therefore, the location of BBP can be confirmed as $K_{BBP} = 1$ & $T_{BBP} = 24$

Table 5-1 Parameters configuration of an ROI

Work-station Index (k^{th})	Processing Time ($T_{p,k}$)	Transfer Time (T_k)	Minimum Gap (T_g)	Inter-Arrival Time (T_e)
1	4	30	2	Case 1: 6 Case 2: Random
2	5	26		
3	3	20		
4	1	40		
5	2	34		
6 (Outlet)	-	20		

Another core function is the component outlet time prediction function *Get_Comp_Outlet_Time()*. In this study, five components were calculated by hand to test the accuracy of this interface. In the case of constant inter-arrival time of 6 time units, it is quite obvious that all three components can flow through the ROI without any delay as the inter arrival time (6 time units) is larger than the process time of the bottleneck work-station (5 time units). When components arrive at the inlet with random inter-arrival time, the output times can be predicted according to Equation (1). The detailed calculation of the five components is illustrated as follows.

First of all, the ideal throughput time (T_s) and process time of bottleneck work-station (T_B) can be obtained easily:

$$T_s = 185 \qquad T_B = 5$$

The 1^{st} component arrives at the clock time 0 ($t_{i,1} = 0$)

As there is no component ahead of it, the total delay of the 1^{st} component is initialized as $\Delta T_{d,1} = 0$ and the predicted outlet time is:

$$t_{o,1} = 0 + 185 + 0 = 185$$

The 2^{nd} component arrives at the clock time 3 ($t_{i,2} = 3$)

The actual inter-arrival time between the 1^{st} and the 2^{nd} component is:

$$T_{e,2} = t_{i,2} - t_{i,1} = 3 - 0 = 3; \quad \text{and} \quad T_{e,2} - \Delta T_{d,1} = 3 - 0 = 3$$

As $2(T_g) < (T_{e,2} - \Delta T_{d,1}) < 5(T_B)$, the value of $\Delta T_{d,2}$ is confirmed and the predicted outlet time of the 2^{nd} component is calculated as follows:

$$\Delta T_{d,2} = T_B - (T_{e,2} - \Delta T_{d,1}) = 5 - 3 = 2$$

$$t_{o,2} = 3 + 185 + 2 = 190$$

The 3^{rd} component arrives at the clock time 5 ($t_{i,3} = 5$)

The actual inter-arrival time between the 2^{nd} and the 3^{rd} component is:

$$T_{e,3} = t_{i,3} - t_{i,2} = 5 - 3 = 2; \quad \text{and} \quad T_{e,3} - \Delta T_{d,2} = 2 - 2 = 0$$

As $\left(T_{e,3} - \Delta T_{d,2}\right) < 2(T_g)$, the value of $\Delta T_{d,2}$ complies with the last branch of $\Delta T_{d,\beta}$ definition and the predicted outlet time of the 3^{rd} component is obtained:

$$\Delta T_{d,3} = t_{o,2} + T_B - \left(t_{i,3} + T_s\right) = 190 + 5 - (5 + 185) = 5$$

$$t_{o,3} = 5 + 185 + 5 = 195$$

The calculations of the 4^{th} and the 5^{th} component are also listed:

The 4^{th} component arrives at the clock time 16 ($t_{i,4} = 16$)

$$T_{e,4} = t_{i,4} - t_{i,3} = 16 - 5 = 11; \quad \text{and} \quad T_{e,4} - \Delta T_{d,3} = 11 - 5 = 6$$

$$\Delta T_{d,4} = 0$$

$$t_{o,4} = 16 + 185 + 0 = 201$$

The 5^{th} component arrives at the clock time 18 ($t_{i,5} = 18$)

$$T_{e,5} = t_{i,5} - t_{i,4} = 18 - 16 = 2; \quad \text{and} \quad T_{e,5} - \Delta T_{d,4} = 2 - 0 = 2$$

$$\Delta T_{d,5} = t_{o,4} + T_B - \left(t_{i,5} + T_s\right) = 201 + 5 - (18 + 185) = 3$$

$$t_{o,5} = 18 + 185 + 3 = 206$$

The functions were invoked in the Microsoft Excel program and the results are displayed in Figure 5-1. First the system configuration parameters were read from the data file. Then after the components arrival times were input, the predicted outlet times of the components were obtained directly by clicking the “Run Prediction” button. The

outputs were exactly in line with the results of the hand simulation, which means that the functions were implemented correctly, as expected.

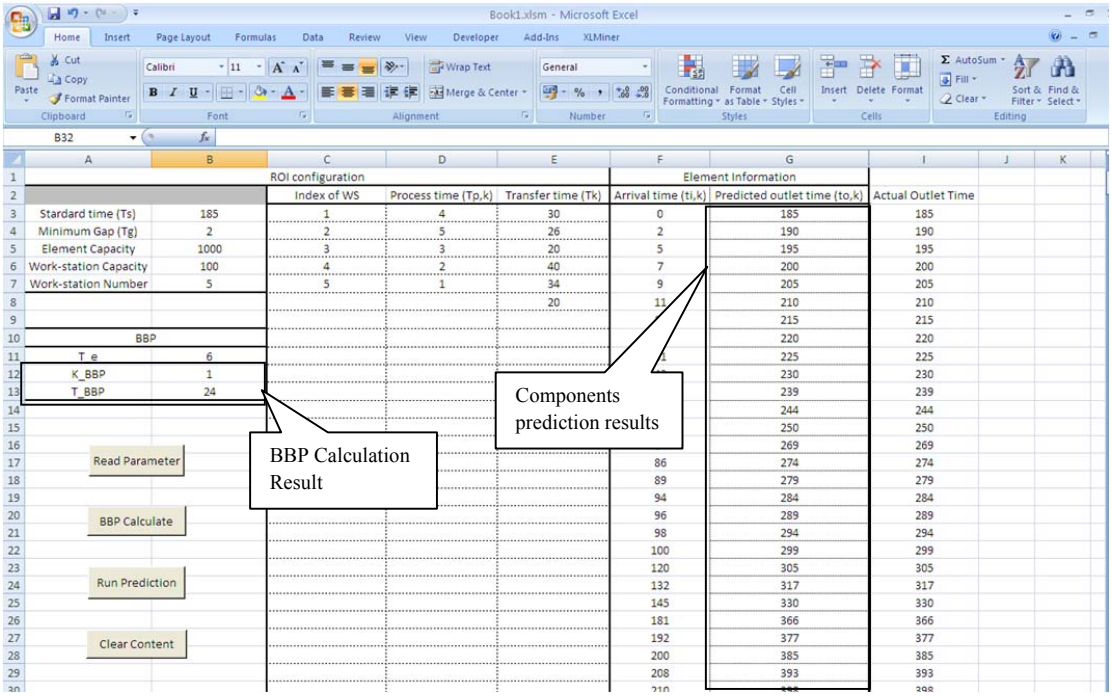


Figure 5-1 Output of functions invoked in Excel program

5.2 Scenarios test within single ROI

This section describes the employment of the production line introduced in Section 5.1 (see Table 1) as an ROI to examine the proposed monitoring methodology. Two types of defect scenarios within the ROI, blocking and slowdown, were analyzed and, for each defect, the analysis process was divided into two steps. First, a simple case with components arriving at a constant interval was used to illustrate the working. Then, the components' arrival pattern was modified to a Poisson distribution to validate the transformation of ROI "pulses" in abnormal situations. The simulation software Arena was selected as the tool to verify the results.

5.2.1 Blocking cases with constant components IAT (T_e = 6 time units)

For blocking cases, Table 5-2 gives the blocking points settings used in the experiments. In the simulation, blocking occurred when the 50th component arrived at the blocking point. Figure A-1 in Appendix A shows the Arena layout in which Blocking Point 1 and Blocking Point 2 represent the BPs before and after the BBP respectively. Detailed information on the simulation can be found in the description of Experiment One in Appendix A.

Table 5-2 Blocking points setting

Blocking Label	After Work-station (the mth work-station)	Time Length from the Work-station (time unit)
BP 1	Inlet	15
BP2	3rd	20

According to the calculation in section 5.1, the Balanced Blocking Point (BBP) of the current ROI is located at 24 time units after the 1st work-station. It should be noted that, when blocking occurred at BBP, the IAT and ILT charts halted simultaneously while the number of the instant WIPs remained constant. The next stage of the study was to map the BP locations against the ΔT_{io} graph, as shown in Figure 5-2, and the calculated maximum response time worked out to be 122 time units according to Equation (8). Supposing, at a certain time, the WIP started declining (IAT halted simultaneously), as in Figure 5-4(a). Immediately, we confirmed that blocking occurred before BBP. Along with the reduction of WIP the potential region of the BP could be narrowed gradually, approaching the Inlet from the BBP. For example, the WIP was still decreasing at time 420 where ΔT_{io} just passed the value -84 (336-420=-84), at that time one could assert that the blocking should occur upstream of the 1st work-station, that is, between the Inlet and the 1st work-station in this case (Also see Figure 5-2). In Figure 5-

3(a), it can be seen that the WIP changing stopped at 468 (ILT halted simultaneously) and hence, the ΔT_{io} arrived at its destination value -132 (336-468=-132) where the BP was eventually able to be pinpointed between the Inlet and the 1st work-station by using the graph in Figure 8. In the situation in Figure 5-3(b), the BP was after the BBP. The value of WIP was increasing from clock time 461 to 582; that is, the final ΔT_{io} was 121 (582-461=121). Referring to Figure 5-2, the BP should have been somewhere between the 3rd and the 4th work-station.

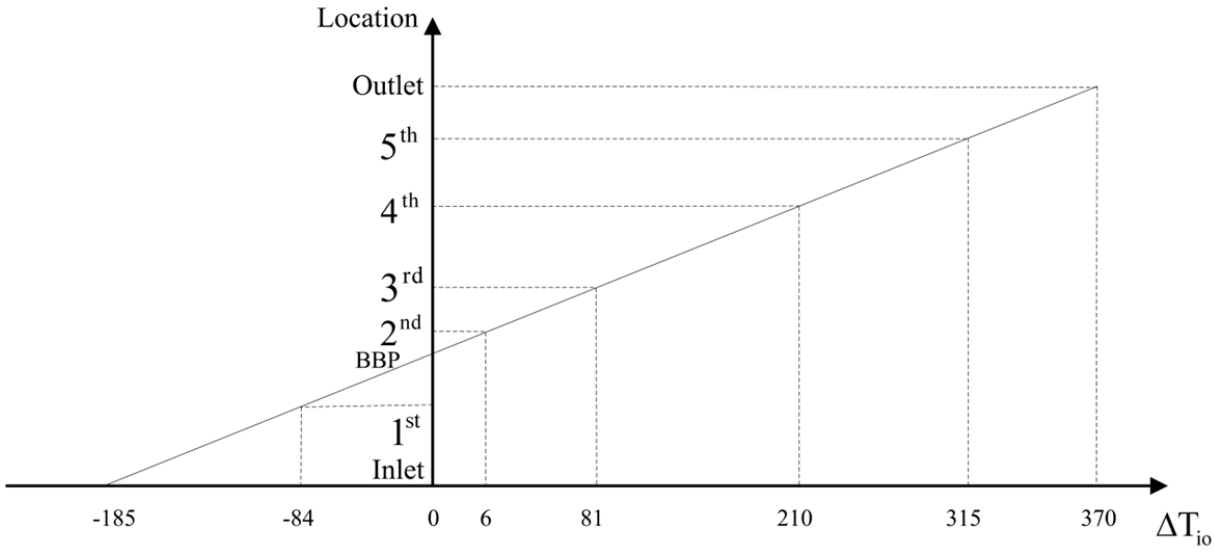


Figure 5-2 Mapping of BP locations onto ΔT_{io} (5 work-stations case)

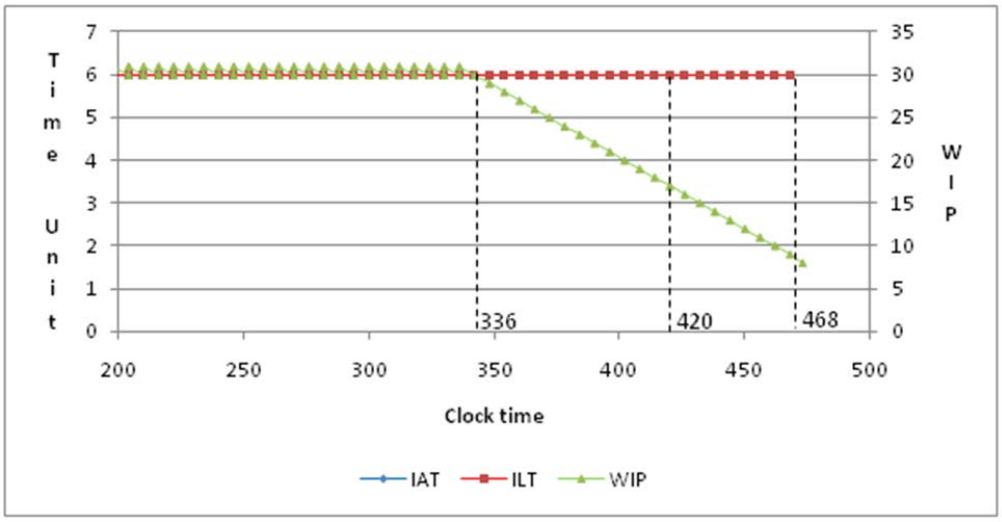


Figure 5-3(a) WIP in production declined (T_e = 6 time units)

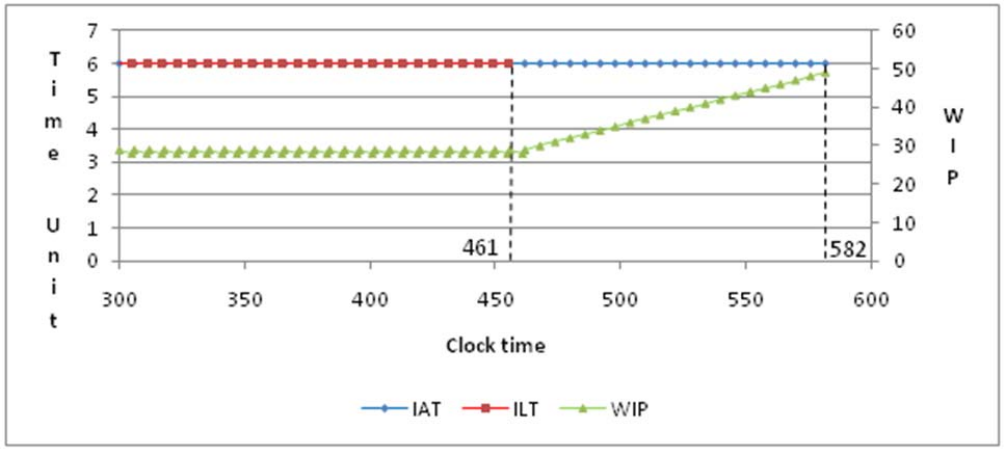


Figure 5-3(b) WIP in production inclined (T_e = 6 time units)

5.2.2 Blocking cases with random components IAT (T_e = Expo(6) time units)

Constant components inter-arrival time is an ideal situation and the proposed method will be more widely applicable if it can cater for a random inter-arrival time; of course, the mean time of the inter-arrival time should always be larger than the processing time of each work-station. Without losing generality, the components are supposed to arrive at the Inlet in a Poisson process so that the inter arrival time between components complies with an exponential distribution (with a mean of 6 time units).

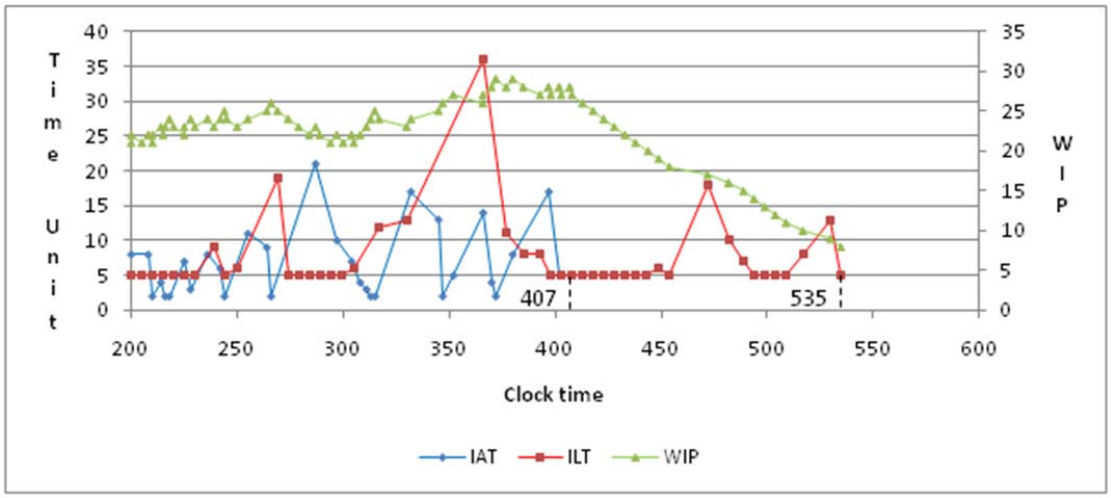


Figure 5-4(a) WIP in production declined (random T_e)

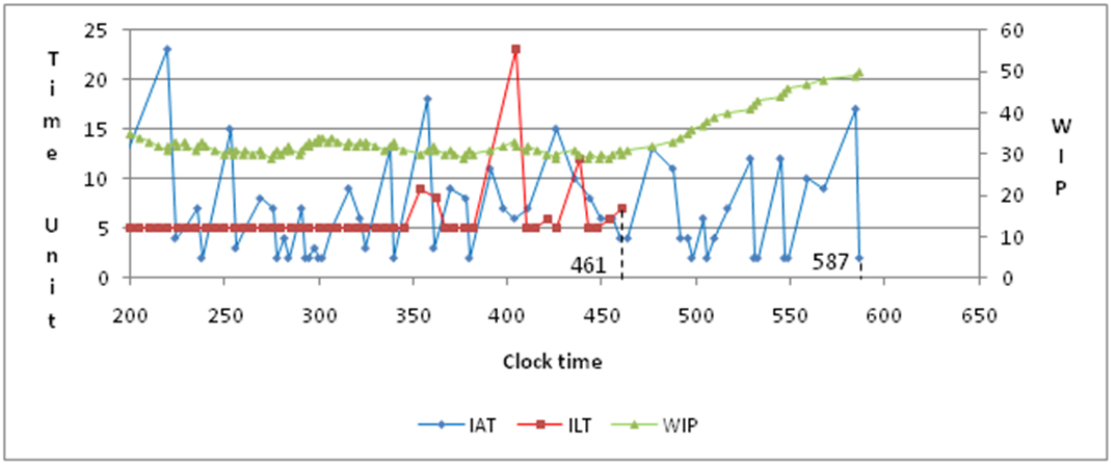


Figure 5-4(b) WIP in production inclined (random T_e)

Figures 5-4(a) and 5-4(b) show the graphs of the same BPs as in Section 5.2.1 under random inter-arrival time, and the experimental results data can be found in Appendices II and Ⅲ. In Figure 5-4(a), IAT and ILT halted at 407 and 535 clock times respectively and ΔT_{io} was therefore obtained as -128 (407-535=-128), which means BP occurred between the Inlet and the 1st work-station with reference to the graph in Figure 5-2; here the observed WIP change may not be rapid due to the random nature. Similarly, in Figure 5-4(b), ILT stopped at 461 and IAT stopped at 587, so ΔT_{io} was 126 (587-

461=126), implying that the BP is between the 3rd and the 4th work-stations. In comparison, these results are slightly different from those of the ideal case as shown before. This has mainly been caused by the variations in confirming the signal from both the IAT and ILT, as there are gaps between two adjacent components.

5.2.3 Slowdown cases with constant components IAT (T_e = 6 time units)

In the next two sub-sections, the simulated slowdown cases are described and the proposed **FLFA** is presented in detail to show how it can be followed to monitor different types of slowdown. Table 5-3 provides the predefined slowdown settings in the experiments. For convenience, all machine slowdowns were considered to have occurred when the 50th component arrived at the target work-station and all transfer slowdowns when the 50th component left the sub-branch shown in Table 5-3. Based on the proposed **FLFA** approach, the first step was to calculate the maximum tolerance of the throughput time enlargement (T_t) based on Equation (14) as:

$$T_t = MAX\left[\left(T_e - MIN\left(T_{p,i}\right)\right), \left(\frac{T_e}{T_g} - 1\right) T_e\right] = MAX[6-1, (3-1)*6] = 12$$

Table 5-3 Slowdown cases settings

Simulation Scenarios	Failure Location	Original Production Time/Transfer time	Altered Production Time/Transfer time
Light Work-station Slowdown	Work-station 4	1	5
Serious Work-station Slowdown	Work-station 4	1	8
Slight transfer slowdown	W3 -> W4	40	80
Serious Transfer slowdown	W3 -> W4	40	160

When components arrive at the Inlet with an IAT equal to 6 time units, RI is zero and ILT is identical to IAT; see Figure 5-5(a). Figure 5-5(b) shows the sudden

enlargement of RI_{50} by the rate of 4. As the amplitude is smaller than the maximum tolerance of the throughput time enlargement ($T_t = 12$), the algorithm goes to **Layer 2**. With RI_{51} keeping a value of 4 and the ΔILT_{51} returning back to 0, the slowdown was a "Slight Machine Slowdown". In Figure 5-5(c), it can be seen that RI_{50} also encountered a sudden increase (13-6=7) and the RI value of the following components kept increasing. From Equation (15), the observation time duration is calculated ($T_{ts} = \left[\frac{ILT_{51}}{T_e} * MAX(T_k)\right] = \left(\frac{8}{6} \times 40\right) = 54$). Up to 56 time units later when the 58^{th} component flowed out of the line, the RI value was still increasing and ILT had a constant value of 8. Therefore from **Layer 4**, there was a "Serious Machine Slowdown" in the production line.

For Figure 5-5(d), the RI value stabilized at 41 and ΔILT_{58} resumed back to 0 within 78 time units; the time duration was still within the maximum observation time duration ($T_{ts} = \left[\frac{ILT_{51}}{T_e} * MAX(T_k)\right] = \left(\frac{12}{6} \times 40\right) = 80$). The problem was thus a "Slight Transfer Slowdown". As in Figure 5-5(e), it can be seen that RI_{50} suddenly leaped to 15 and was much larger than the maximum tolerable throughput time enlargement ($T_t = 12$) so there was a "Serious Slowdown" and one can go from **Layer 1** to **Layer 4** for a further judgment. As ILT_{58} decreased to 8 within the maximum observation time duration ($T_{ts} = \left[\frac{ILT_{51}}{T_e} * MAX(T_k)\right] = \left(\frac{24}{6} \times 40\right) = 160$) rather than keeping the value of 24, it was a "Serious Transfer Slowdown".

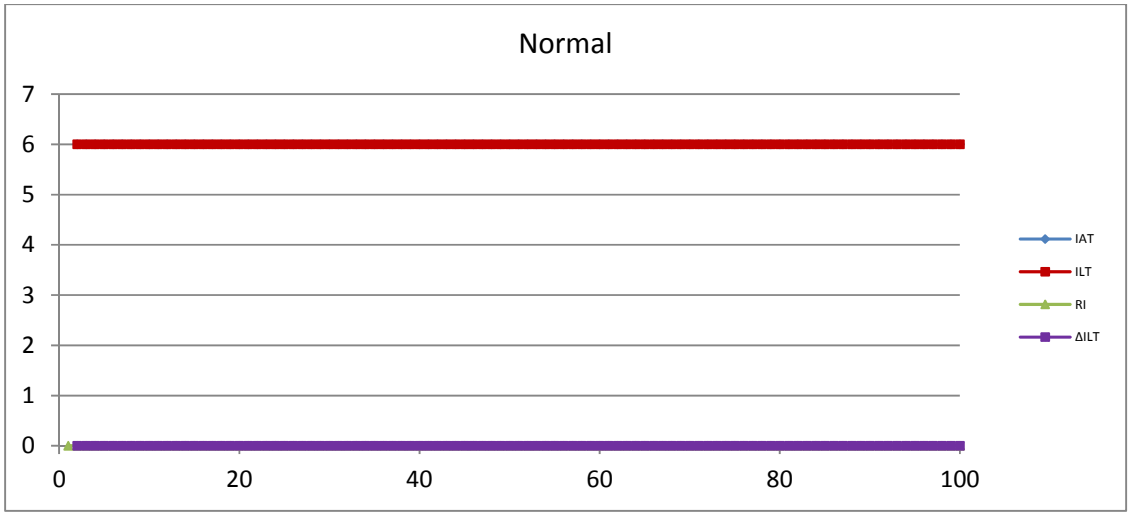


Figure 5-5(a) Normal operation (constant inter-arrival time)

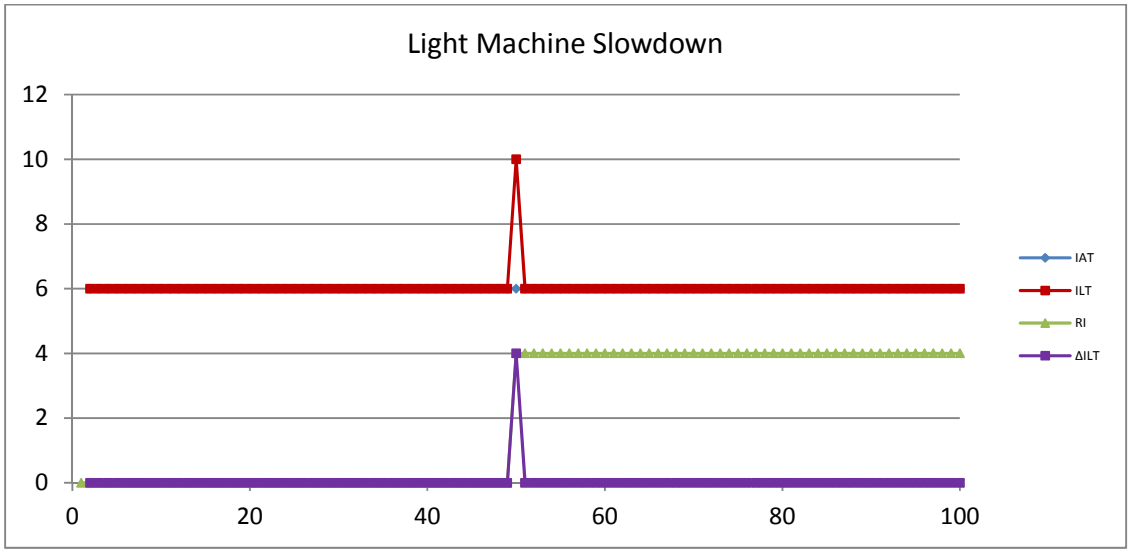


Figure 5-5(b) Slight machine slowdown (constant inter-arrival time)

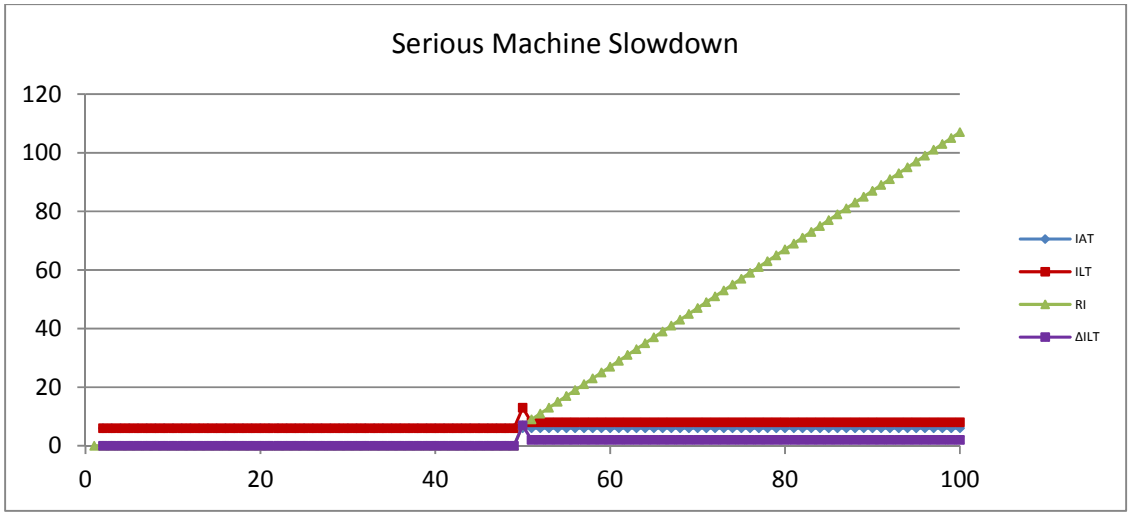


Figure 5-5(c) Serious machine slowdown (constant inter-arrival time)

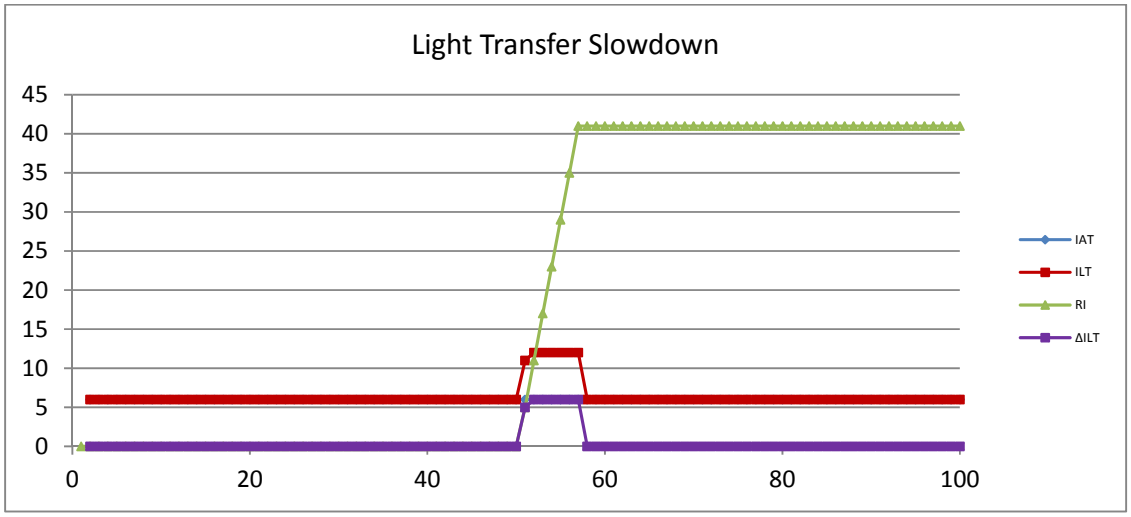


Figure 5-5(d) Slight Transfer Slowdown (constant inter-arrival time)

Figure 5-5(e) Serious transfer slowdown (constant inter-arrival time)

5.2.4 Slowdown cases with random components IAT (Expo(T_e)= Expo(6) time units)

Similar to the blocking cases experiments, components are supposed to arrive at the Inlet in a Poisson process to test the effectiveness of the FLFA approach on identifying slowdown symptoms.

Figure 5-6 presents graphs of the same slowdowns as in Case 1 under an exponential arrival interval. Despite the IAT and the ILT fluctuations that were affected by their exponential behaviour, both the ΔILT and RI values remained at zero under a normal status (see Figure 5-6(a)). In Figure 5-6(b), RI_{50} suddenly jumped to 4, that is smaller than $T_t = 12$; with ΔILT_{51} returning to 0 and RI_{51} keeping a value of 4, the slowdown was a "Slight Machine Slowdown". In Figure 5-6(c), RI increased from the 50^{th} component and did not stop within the maximum observation time duration ($T_{ts} = \left(\frac{9}{6} \times 40\right) = 60$) so it was a "Serious Slowdown" (leave **Layer 3** and go to **Layer 4**). Since ILT stayed at 8 within T_{ts}, it was a "Serious Machine Slowdown". Figure 5-6(d)

shows that after the temporary increase of RI, it stabilized at 14 within the maximum observation time duration ($T_{ts} = \left(\frac{5+6}{6} \times 40\right) = 74$) and then the ΔILT returned to 0, so it was identified as a "Slight Transfer Slowdown" (**Layer 3**). In Figure 5-6(e), it can be seen that RI_{50} suddenly leaped to 30, much greater than $T_t = 12$, and the algorithm directed a move to **Layer 4**. As the ILT values decreased from 20 to 8 within the maximum observation time duration ($T_{ts} = \left(\frac{15+6}{6} \times 40\right) = 140$), there was no doubt that a "Serious Transfer Slowdown" had occurred (**Layer 4**).

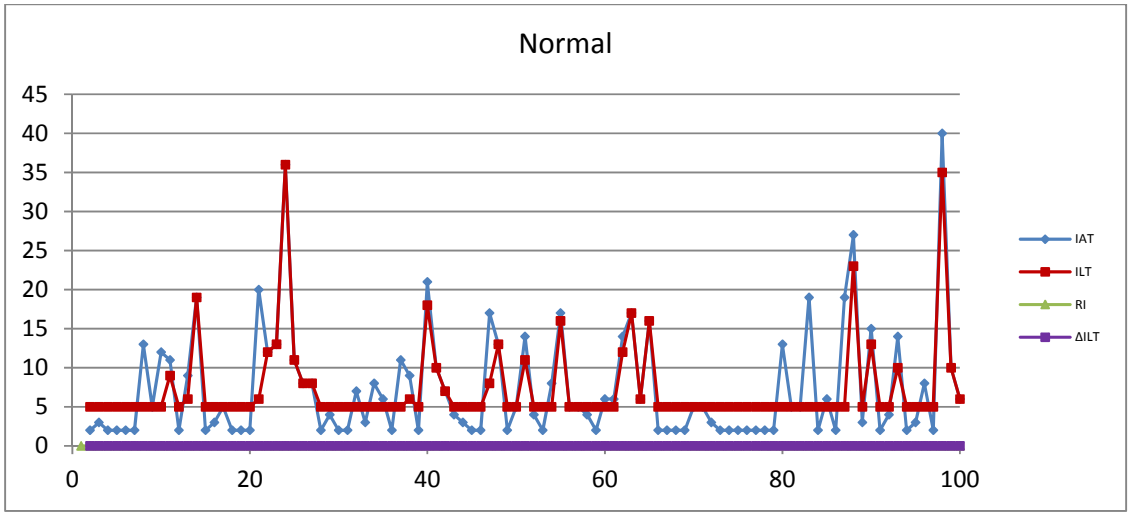


Figure 5-6(a) Normal operation (exponential inter-arrival time)

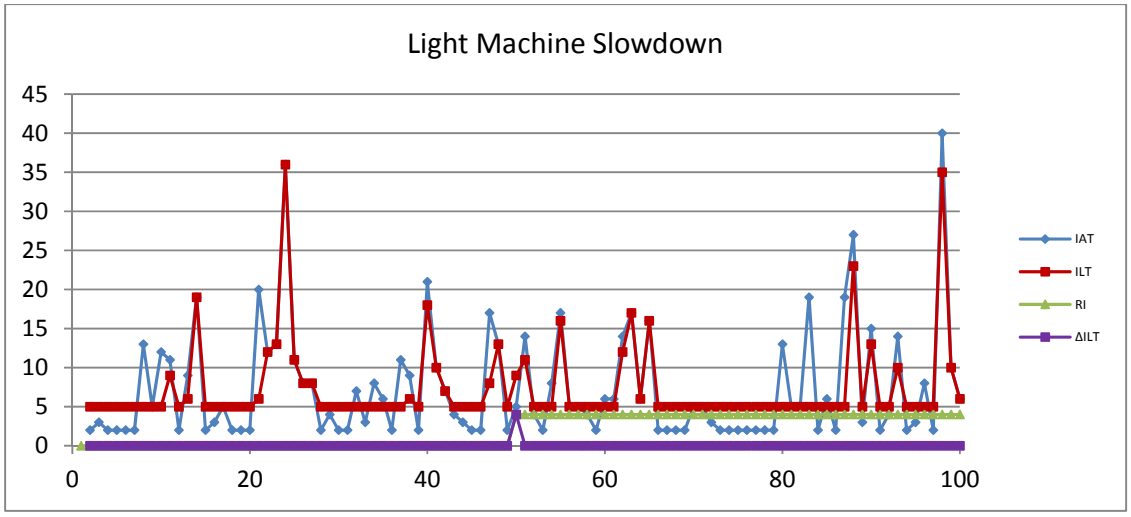


Figure 5-6(b) Slight machine slowdown (exponential inter-arrival time)

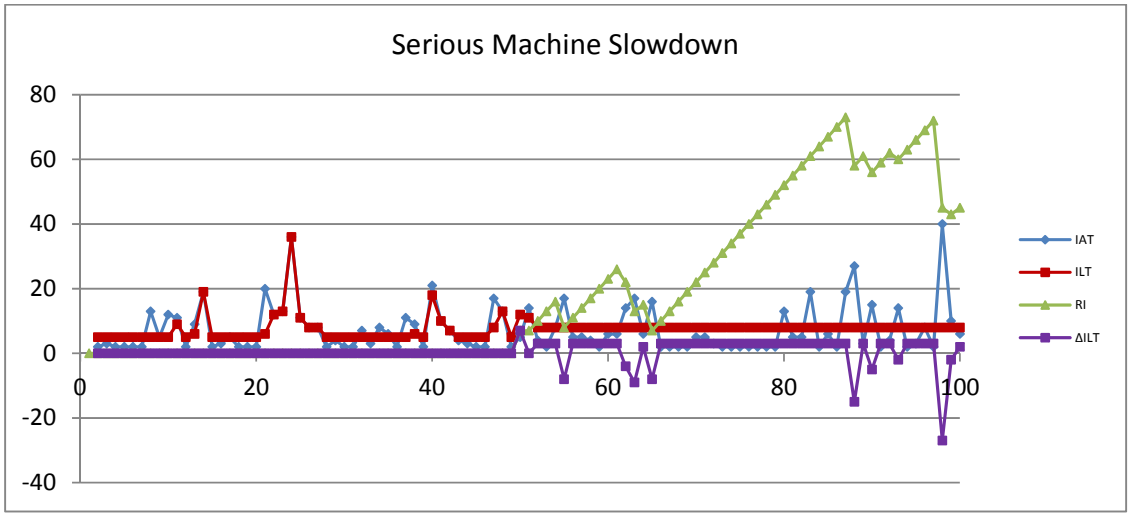


Figure 5-6(c) Serious machine slowdown (exponential inter-arrival time)

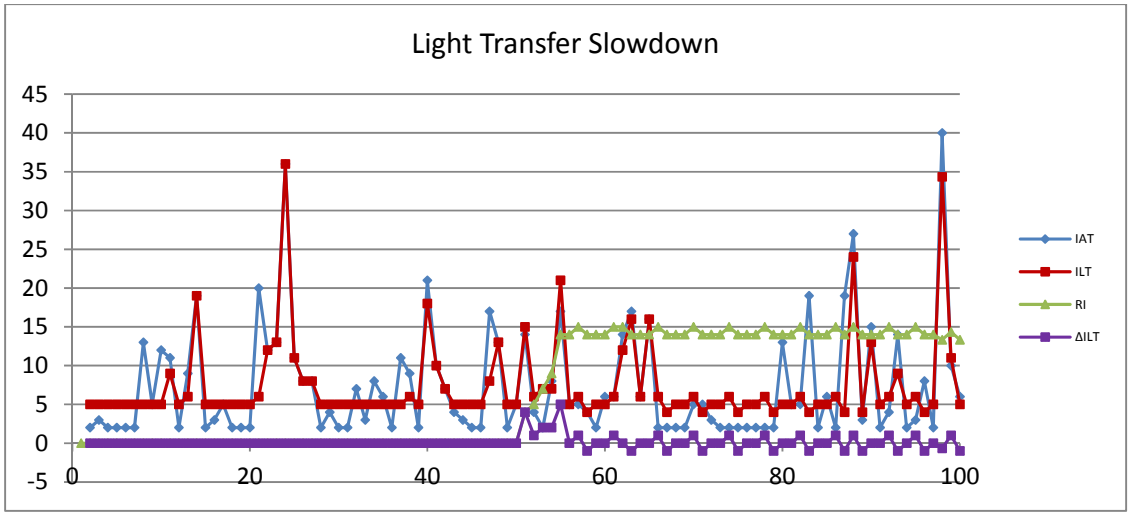


Figure 5-6(d) Slight transfer slowdown (exponential inter-arrival time)

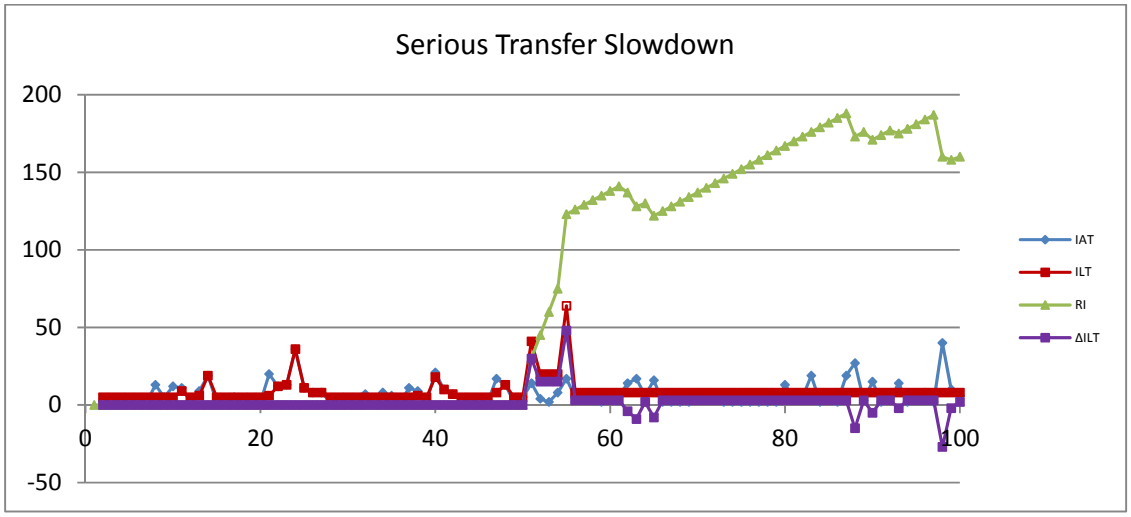


Figure 5-6(e) Serious transfer slowdown (exponential inter-arrival time)

5.3 Monitoring framework design by means of ROI segmentation

A virtual long production line with 15 work-stations connected by conveyors was employed here to validate the monitoring framework design methods. Table 5-4 summarizes the production line configuration, with the transfer time between work-stations set at 10 time units (a constant value). With the given configuration parameters,

the relationship between the maximum response time and the length of the production line (in time unit) was first established and the **3-Step ROI Segmentation** technique illustrated in Section 3.7 was implemented in detail. Then, a series of simulation experiments was conducted to examine the response times of various defects in the production line to test the effectiveness of the proposed monitoring methodology.

Table 5-4 Configuration of a long virtual production line

Index of work-stations (k^{th})	Processing Time ($T_{p,k}$)	Transfer Time (T_k)
0 (Inlet)	-	-
1	6	10 (from Inlet)
2	7	10
3	8	10
4	5	10
5	6	10
6	4	10
7	7	10
8	6	10
9	5	10
10	3	10
11	5	10
12	6	10
13	7	10
14	4	10
15	3	10
16 (Outlet)	-	10 (To Outlet)

*5.3.1 Calculating tolerable maximum response time of blocking case [**Step 1**]*

In the experiments, the tolerable maximum response time was set as 45 time units ($T_{r,c} = 45$) and the inter-arrival time of components and the minimum gap between components were 8 and 4 time units respectively ($T_e = 8, T_g = 4$). The severity rate of the slowdown case was taken as double the normal inter-arrival time ($T_{SD} = 16$). Accordingly, the tolerable maximum response time of the blocking case was calculated from Equation (23) as 33 time units. In addition, to remove the effect due to the inter-

arrival time between parts on the response time, the actual tolerable maximum response time of the blocking case was subtracted by T_e and became 25 time units ($\overline{T}_{rb,max} = 25$).

*5.3.2 Response time on different T_e and T_g combinations [**Step 2**]*

For this long virtual production line, the relationship between the maximum response time of the blocking case ($T_{r,b,max}$) and the length of the line (T_s) has been worked as in the 3rd line in Figure 5-7; other lines in Figure 5-7 were plotted to exhibit the relationship under different scenarios. It is easy to understand that to keep the production line healthy, the inter-arrival time (T_e) should always be larger than (or equal to, at most) the processing time of the bottleneck work-station ($T_e \geq MAX(T_{p,k})$).

When the gap between two parts (T_g) is small in comparison to T_e, the maximum response time of the blocking case ($T_{rb,max}$) will approach the ceiling at which the main slope equals 1, such as the upmost line in Figure 5-7. This actually describes the situation of an unlimited-buffer transfer line and blocking can only be detected at the Outlet when there are no more components flowing from the output. In contrast, when the value of T_g approaches T_e, the response time immediately becomes ($T_{r,b,max} \rightarrow 0$) no matter how lengthy an ROI is. This represents another extreme condition where the parts enter an ROI with little time gap in between. Once blocking occurs, the abnormality can be detected almost instantly at the Inlet as no more parts can be fed into the line. For the other combinations of T_e and T_g, the curves governing $T_{r,max}$ with respect to T_s keep increasing linearly with the addition of some sudden changes, which are caused by the migration of BBP as T_s increases. The slope of the linear lines is determined by $(1 - \frac{T_g}{T_e})$);

that is to say, without considering the impact of the migration of BBP the smaller the $(\frac{T_g}{T_e})$, the larger the maximum response time obtained. See also the two cases: $(T_e = 8, T_g = 2)$ and $(T_e = 8, T_g = 4)$.

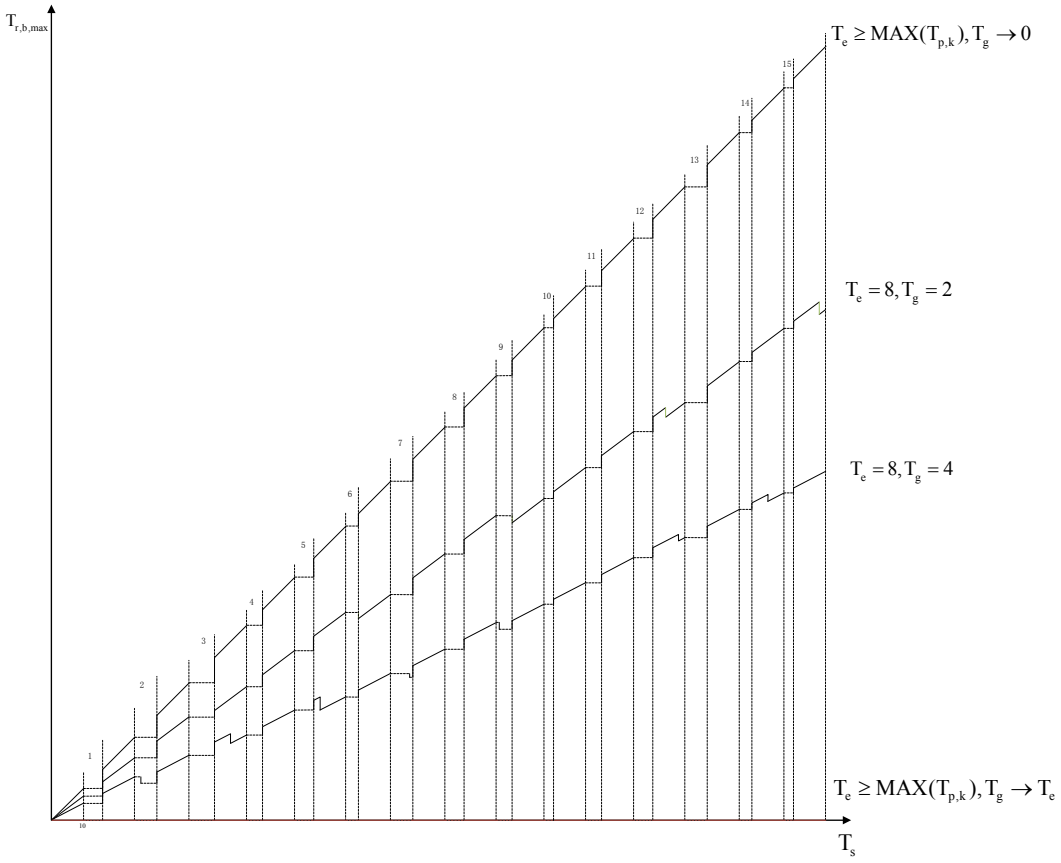


Figure 5-7 Relationship between $T_{r,max}$ and T_s with different combinations of T_e and T_g

*5.3.3 ROI segmentation subjected to the specified response time [**Step 3**]*

According to the previous two steps, the point where $\overline{T}_{rb,max} = 25$ on the $T_{rb,max}$ against T_s graph is of interest. Starting with $K_{BBP} = 0$, with the given T_e and T_g values, the graph of $T_{r,max}$ against T_s can be plotted. It is noted that with the increase of the potential ROI length, the position of BBP gradually migrates towards the outlet; each time when the

length of ROI arrives at a value of $K_{BBP} * T_e + \frac{T_e}{T_g}\sum_{k=0}^{K_{BBP}} T_k$, the BBP encounters migration across a work-station (the value of K_{BBP} increases by 1) and the plot of $T_{r,b,max}$ against T_s drops down with a amplitude of $\sum_{k=0}^{K_{BBP}} T_{p,k} - K_{BBP} * T_g$ in comparison to that of $K_{BBP} = 0$.

Table 5-5 Detail information for plotting the initial two segments

Segment 1 (ROI1)	$T_s = 242$	
Work-station index before potential BBP (K_{BBP})	Position of jump ($K_{BBP} * T_e + \frac{T_e}{T_g}\sum_{k=0}^{K_{BBP}} T_k$)	Amplitude of jump ($K_{BBP} * T_g - \sum_{k=0}^{K_{BBP}} T_{p,k}$)
1	28	-2
2	56	-5
3	84	-9
4	112	-10
5	140	-12
6	168	-12
7	196	-15
8	224	-17
Segment 2 (ROI2)	$T_s = 182$	
Work-station index before potential BBP (K_{BBP})	Position of jump ($K_{BBP} * T_e + \frac{T_e}{T_g}\sum_{k=0}^{K_{BBP}} T_k$)	Amplitude of jump ($K_{BBP} * T_g - \sum_{k=0}^{K_{BBP}} T_{p,k}$)
1	10	-1
2	38	-3
3	66	-3
4	94	-6
5	122	-8
6	150	-9
7	178	-9

Taking Figure 5-8(a) as an example, when the length of the potential ROI arrives at 28, BBP migrates from between the Inlet and the 1st work-station to between the 1st and the 2nd work-station, and the value of K_{BBP} increases from 0 to 1. At the same time the plot changes by -2 (negative means downward) units due to the effect of the 1st work-station. This $T_{r,b,max}$ Vs T_s graph is plotted in this way up to the entire production line (T_s=242). Details of the first two ROI segments are shown in Table 5-5 and Figure 5-8(a,

b); to reduce complication in these tables, only the major portion of work-stations are presented, but the associated graphs show the whole production line.

In Figure 5-8(a), the horizontal line at $\overline{T}_{rb,max} = 25$ cuts the position between the 3rd and the 4th work-stations so $T_s = 60$ was chosen. Then, the first ROI was taken away from the production line and a similar procedure was applied on the remaining line, such as in Figure 5-8(b). The final result of the whole production line segmentation is shown in Figure 5-9, where 6 counters are needed to serve 5 sub-branches (ROI1 to ROI5) to accomplish the monitoring assignment, subjected to the specified response time, and the lengths of the sub-branches are 60, 57, 57, 61, and 7 consecutively.

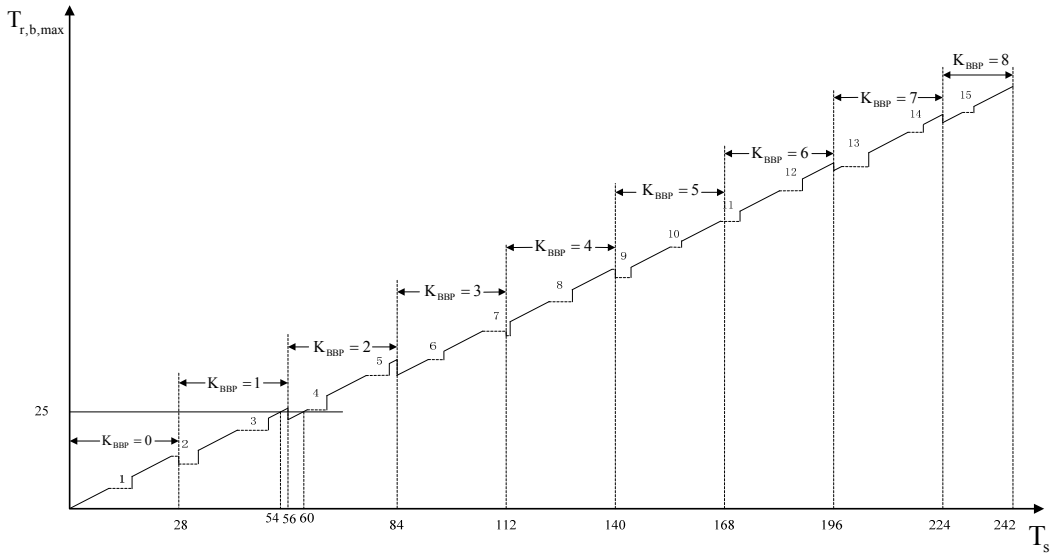


Figure 5-8(a) First ROI segment to fulfill $T_{r,c} = 45$ time units

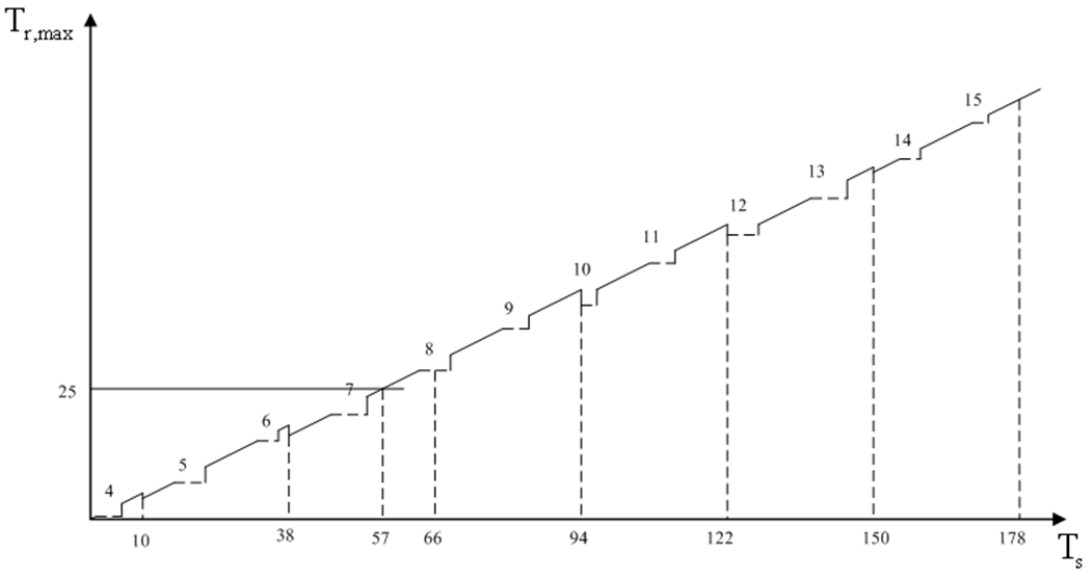


Figure 5-8(b) Second ROI segment to fulfill $T_{r,c} = 45$ time units

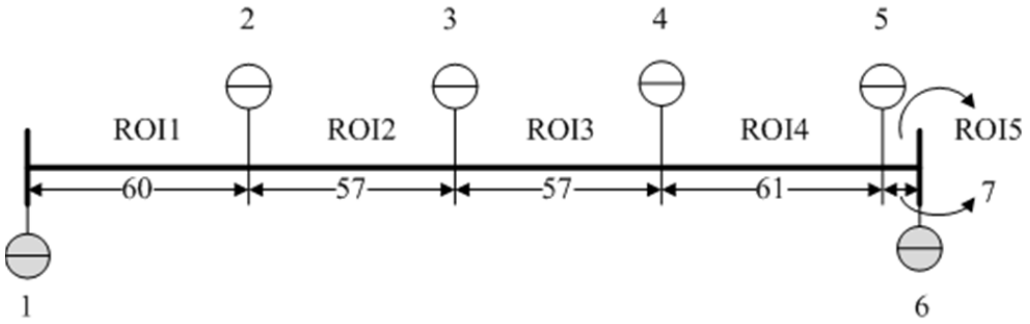


Figure 5-9 Result of ROI segmentation by graph plotting

The results can also be obtained conveniently by invoking the user interfaces implemented in Chapter 4. Figure 5-10 shows the results of the ROIs segmentation in the Excel program. It can be seen that the results are the same as the ones obtained through the above graph-plotting method.

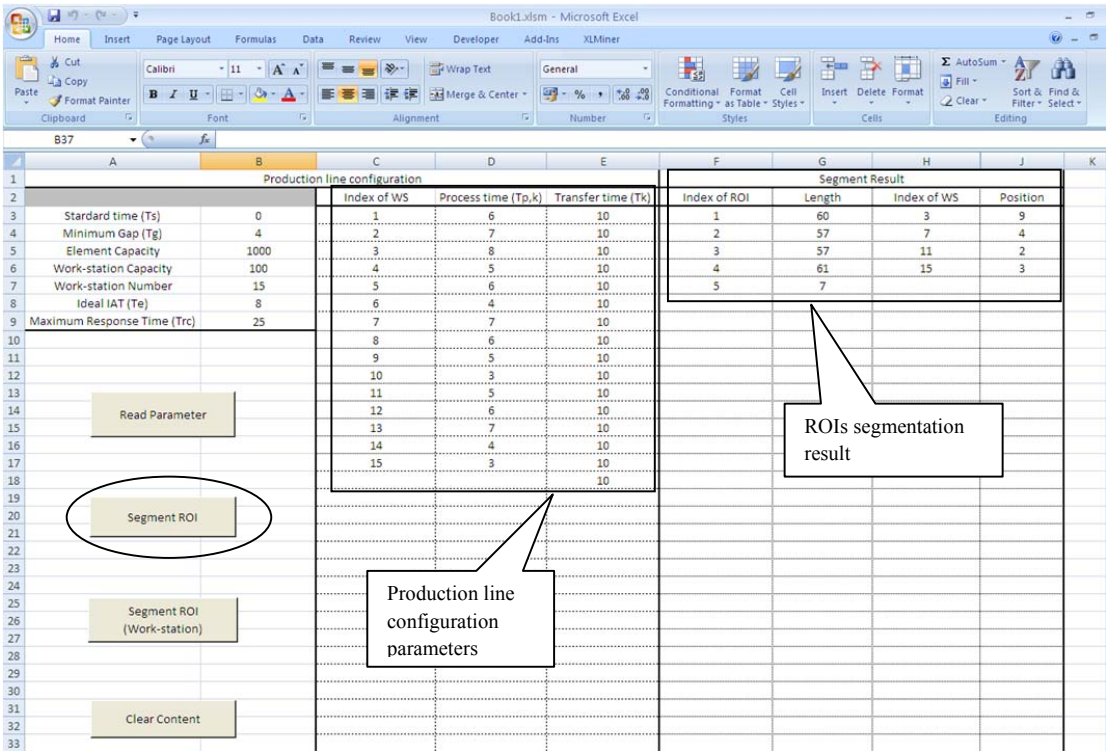


Figure 5-10 Result of ROI segmentation by invoking general interface

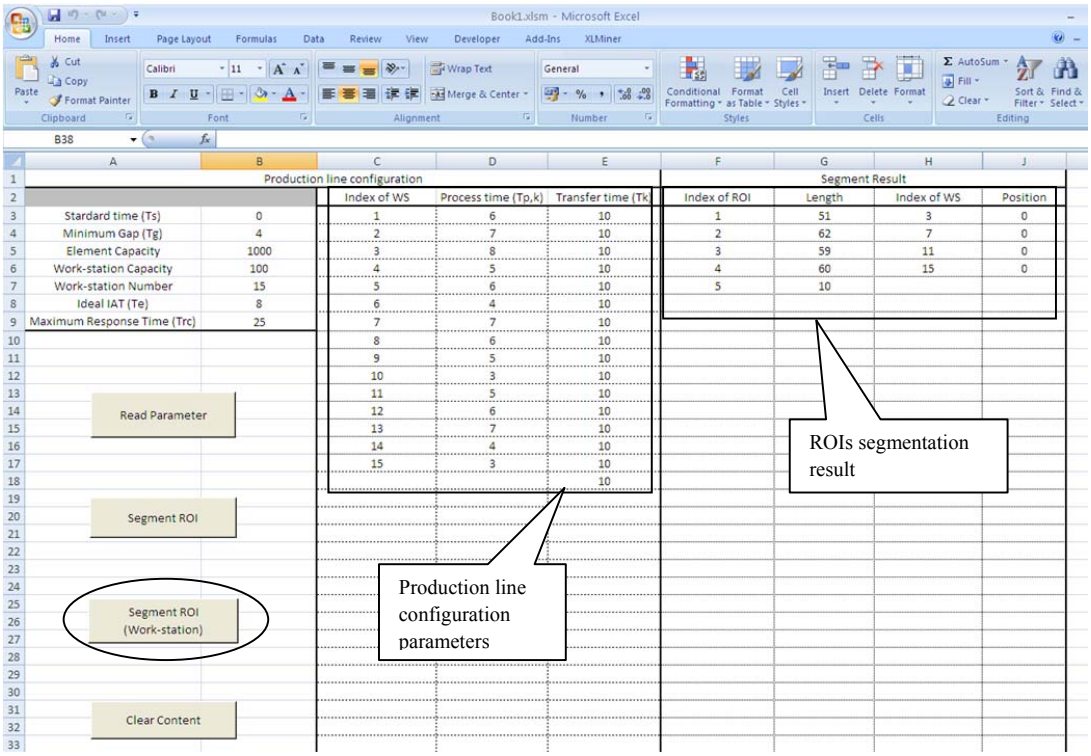


Figure 5-11 Result of ROI segmentation considering segmentation points constraints

Figure 5-11 shows the ROIs segmentation results when considering the constraint that the segmentation points can only be installed at the exit of work-stations. It is not hard to observe that the segmented ROIs in Figure 5-11 are slightly different from the results in Figure 5-10. The contents in the column "Position" are all "0", which means the segmentation points lie directly at the exits of the 3rd, 7th, 11th, and 15th work-stations.

5.3.4 Validation of ROI segmentation result

To validate the proposed method, Arena® was again employed to conduct the simulation experiments. A virtual production line was constructed and segmented into 5 ROIs according to Figure 5-9. Obviously ROI5 works for a short duration only (7 time units) and its response time will never exceed the specified value. The other four ROIs blockings at different locations (as in Figure 5-12) were tested to examine the maximum response times. Details of the experiment can be found in Experiment Three of Appendix A. Blocking and slowdown scenarios were conducted at different locations along the production line and the response times were collected. To make the presentation uncomplicated, the simulation results have been consolidated in Figure 5-12 and the detailed data of "Response Time Vs Location" is attached in Appendix E. It can be seen from Figure 5-12, that although the response times of the faults detection fluctuated along the production line, they never exceeded the specified maximum response time. From the internal viewpoint, the response times of the first four ROIs (see Figure 5-13) increased with increase of fault locations before the peak points (their BBPs) and then decreased gradually along the rest. The maximum response times for all ROIs are all less than 33 time units (the maximum tolerable response time).

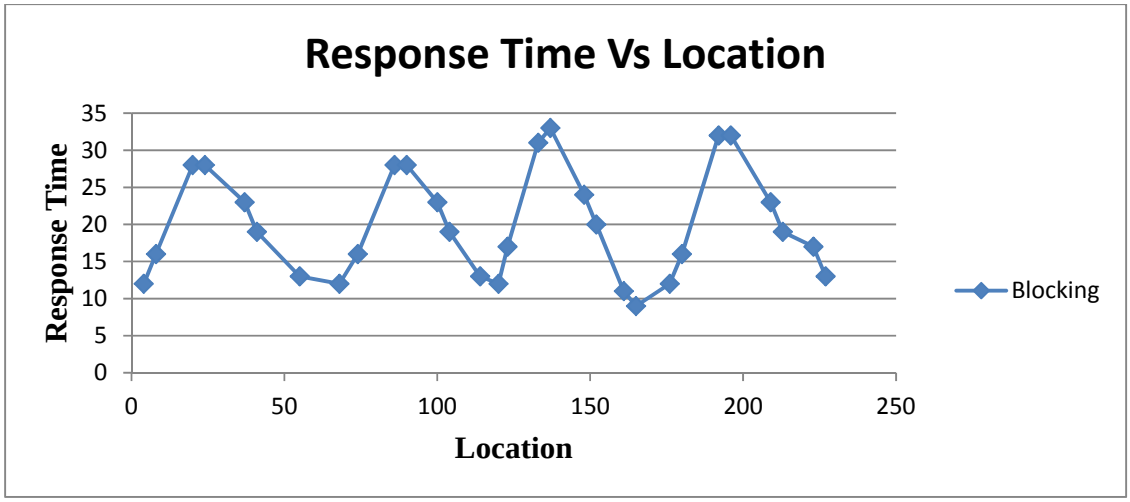


Figure 5-12 Graphs of response time against location along the whole line

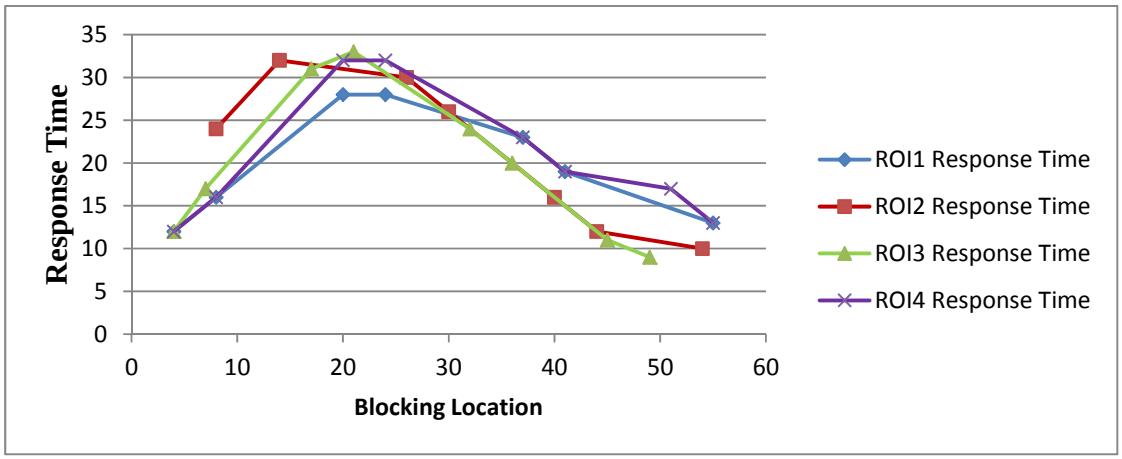


Figure 5-13 Graphs of response time against location within separate ROIs

Chapter Six - Discussion and future work

Although the effectiveness of the proposed methodology has been validated through simulation cases, it is also of great concern how the methodology compares with other similar monitoring approaches. In addition, it is possible to encounter different variances when applying the proposed methodology in practical situations. In the following sections, the comparison among the proposed monitoring method, CIM monitoring method, and ANDON is first stated. Then, the warm-up period in relation to the component leaving time prediction is discussed. And then some crucial steps in adopting the proposed methodology in a multiple products production line are elaborated and the potential future work direction is also examined. Finally, one of the possible alternative applications of the proposed model is presented.

6.1 Comparison with previous works

An ROI-based monitoring methodology is introduced in this research. This methodology enables the holistic supervision of an automated production line with a low-cost investment and convenient implementation. As a coin has two sides, the proposed method has its own pro and con. Compared with related existing works, say CIM and TPS monitoring approaches, the major advantages of ROI-based monitoring methodology are as follows.

First, the proposed monitoring system design requires less people involvement and the implementation process is more straightforward, so that the success rate of the application can be improved. The design of both CIM and TPS approaches requires adequate employee input and sufficient support in the decisions about operational

changes. Otherwise the serious consequence may be system incompatibility and resistance to the new system. In other words, the design process of the proposed monitoring method is much less time-consuming than CIM and TPS monitoring approaches.

Second, the proposed monitoring method presents the holistic health situation of the target system in a straightforward way, therefore reducing the requirements on workers' skills. On the contrary, the success of TPS highly relies on the experience of the employees and the operators in the CIM system should also be of high technical skills.

Third, cost saving is one of the major advantages of the ROI-based monitoring method. Unlike CIM system demanding high investment in both stand-alone equipment and integrated software development or TPS system requiring great expense on employee training and customized equipment research and development, the proposed monitoring method can be implemented only with simple counter devices with time stamp functions. In addition, the extension and migration of the system require much less extra disbursement.

Having said the benefits of the proposed ROI-based monitoring method, there are also several limitations. First, the method is proposed for line-type production system and may not be appropriate for some other types of manufacturing system, such as shop floor production. Second, the sensitivity of the proposed monitoring method depends on the segmentation of ROIs. In certain cases, even though the design of the proposed monitoring method meets the tolerable maximum response time requirements, the actual response time maybe a little longer than the CIM or TPS system. Third, though the

healthiness of the whole production can be supervised by the combination of single ROIs in the system, the interrelationship among the ROIs is not studied yet; therefore the impact of certain ROI abnormality on its upstream and downstream ROIs remains unexplored.

6.2 ROI warm-up period

For an ROI, the component leaving time can be predicted through the proposed technique (see Equation (1)). That is to say, to guarantee the prediction accuracy, the process should always start at a component with no impediment from its instant upstream component. Obviously an empty ROI is undoubtedly eligible for this prerequisite, but the requirement of an empty ROI to cater for the need of the monitoring methodology at the starting stage is not always practical during the operation of a manufacturing system. For example, the restarting of a production line after a temporary shutdown is a typical case of a non-empty ROI. To apply the proposed model in such a situation, a "warm-up" period should be introduced.

Suppose there are N components in the ROI and the indexes of the following components entering the ROI are N+1, N+2, and so on. The clock times when components arrive at the inlet and leave from the outlet are recorded. Instead of being calculated using Equation (1), the accumulated time delay of the $(\mathrm{N}+1)^{\mathrm{th}}$ component is set as its leaving time subtracted by both the arrival time and the ideal throughput time of the ROI such as:

$$\Delta \mathrm{T}_{\mathrm{d,N+1}} = t_{\mathrm{o,N+1}} - t_{\mathrm{i,N+1}} - \mathrm{T_s}$$

Actually $\Delta T_{d,N+1}$ incorporates the accumulated time delay before and with this as the initial value, the leaving time of the subsequent components can again be predicted by Equation (1). In fact, the time duration from the $(N+1)^{th}$ component arriving at the inlet to it leaving the ROI can be regarded as the "warm-up" period. To ensure the reliability of the monitoring process, a "warm-up" period should be employed before the proposed technique starts to work. Besides being applied to start the monitoring of a non-empty ROI, the "warm-up" period is also useful for the resetting of a normal monitoring activity, if needed.

6.3 Adaptability of proposed monitoring method

This research proposes an innovative method of monitoring production using only basic counting devices and the method is considered applicable to most line-type productions. It is beneficial to such kind of manufacturing because it is simply exercised and the line abnormalities can be detected and then located easily. For the other types of production process like cellular manufacturing, the proposed model can also be applicable with little related adjustments.

In the development of the monitoring method, we assume implicitly that only one type of product is involved in the production line. To accommodate multiple types of products is an issue to be addressed. It is anticipated that the proposed technique can also be enhanced to monitor a multiple products system.

6.3.1 Multiple product types production line monitoring

Generally, there is a possibility for a product family with multiple types of products on a production line. Figure 6-1 is a schematic representation of a production line for multiple

product types (Kalir & Arzi, 1997). In such a case, the processing time of a particular work-station in the ROI modelling ($T_{p,k}$) may not be the same at all; it can be a variable according to the product types. Therefore, the time-stamp counters installed at the inlet and the outlet should be further enhanced and equipped with the function of identifying the product type (j), and then linking the processing time of a work-station to the product type ($T_{p,j,k}$). However, it is still incorrect to predict a component leaving time by the proposed model, as the bottleneck work-stations for different product types can be quite different; see Section 3.3 for the involvement of the bottleneck work-station in the model development.

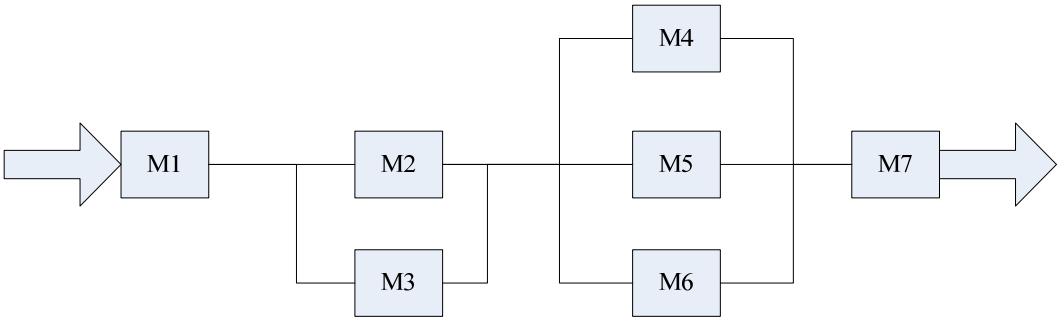


Figure 6-1 Schematic layout of multiple product types production line (Kalir & Arzi, 1997)

One possible alteration is that, rather than calculating the total time delay at the ROI outlet, the new approach tends to work out the time delay before each work-station instead. Suppose that the functional relationship between the type (j) and the index (β) of a component is represented as j = f(β) (see Figure 6-2). The component arrival time at the k^{th} event is affected by the summation of four parts: the clock time at the inlet ($t_{i,\beta}$), the ideal time duration from the inlet to the k^{th} event ($\sum_{i=0}^{k} T_i + \sum_{i=0}^{k-1} T_{p,f(\beta),i}$), the total time

delay from the inlet to the (k-1)th event ($\Delta T_{s,k-1,\beta}$), and the time delay caused by the blocking of the previous component from the (k-1)th event to the kth event ($\Delta T_{b,k,\beta}$), thus:

$$t_{a,k,\beta} = t_{i,\beta} + \sum_{i=0}^{k} T_i + \sum_{i=0}^{k-1} T_{p,f(\beta),i} + \Delta T_{s,k-1,\beta} + \Delta T_{b,k,\beta} \quad (24)$$

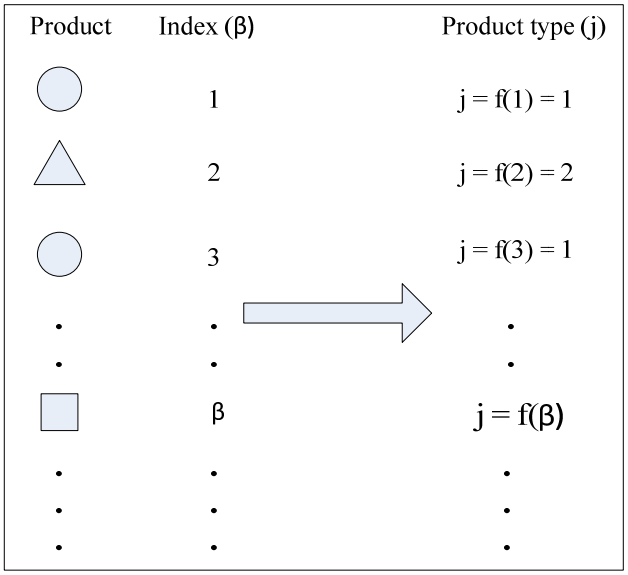


Figure 6-2 Functional relationship between product type and index

As this calculation is carried out progressively one-by-one for all work-stations, the total time delay before leaving from the (k-1)th event is available from the previous iteration. Thus, the last parameter that needs to be determined is only the time delay caused by the blocking of the previous component between the (k-1)th and the kth event. Of course, if there is no blocking caused by the previous component, then:

$$\Delta T_{b,k,\beta} = 0$$

and

$$t_{a,k,\beta} = t_{i,\beta} + \sum_{i=0}^{k} T_i + \sum_{i=0}^{k-1} T_{p,f(\beta),i} + \Delta T_{s,k-1,\beta}$$

(25)

The possible delay before the work-station depends on both the arrival time of the current component and the elapsed time of its immediately previous component. In other words, once the previous component occupies the work-station, the work-station can only return to being accessible again after its processing time. Hence, the possible delay of the current component is determined by:

$$\Delta T_{I,k,\beta} = \max\{T_{k,f(\beta-1),p} - [t_{a,k,\beta} - (t_{a,k,\beta-1} + \Delta T_{I,k,\beta-1})], 0\} \quad (26)$$

The ideal time when the β^{th}component begins to receive service on the k^{th} work-station can be worked out by adding $\Delta T_{I,k,\beta}$ in Equation (25) to $t_{a,k,\beta}$ in Equation (26). By comparing the value with that of the previous component, the actual blocking time delay on the component between the $(k-1)^{th}$ event and the k^{th} event can then be obtained:

$$\Delta T_{b,k,\beta} = \max\left[\left(t_{i,\beta-1} + \sum_{i=0}^{k-1} T_{p,f(\beta-1),i} + \Delta T_{s,k,\beta-1} + T_g\right) - \left(t_{i,\beta} + \sum_{i=0}^{k-1} T_{p,f(\beta),i} + \Delta T_{s,k-1,\beta} + \Delta T_{I,k,\beta}\right), \quad 0\right]$$

(27)

The actual arrival time at the k^{th} work-station can be updated by substituting Equation (27) into Equation (24). The total time delay of the β^{th} component from

arriving at the inlet to passing the k^{th} work-station is stored through Equation (28) for the calculation of arriving at the next work-station:

$$\Delta T_{s,k,\beta} = \Delta T_{s,k-1,\beta} + \Delta T_{I,k,\beta} + \Delta T_{b,k,\beta} \tag{28}$$

In this way, the clock time of the component leaving can be predicted by:

$$t_{o,\beta} = t_{a,n+1,\beta} = t_{i,\beta} + \sum_{i=0}^{n} T_i + \sum_{i=0}^{n} T_{p,f(\beta),i} + \Delta T_{s,n,\beta} \tag{29}$$

With this new ROI and component moving model, the proposed monitoring method can possibly be adopted by a multiple-products production line, but further examination will be needed as this only provides some initial thoughts without an in-depth study to find solid proof at the moment.

6.3.2 Universal monitoring framework

To cover the requirements of a wide range of manufacturing systems and to improve the flexibility of the proposed monitoring methodology, a universal monitoring framework is worth looking into in future research; the preliminary suggestion for this is shown in Figure 6-3. The *System Handling* domain serves as the interface between the physical system and *Component Moving Model* as well as the *Reasoning Kernel* domains. For a manufacturing system, *ROIs* are formed with reference to corresponding response time requirements. With each *ROI*, a *Component Moving Model* and a *Reasoning Kernel* are associated. By doing so, the expansion of the monitoring system will become more well organized, with high flexibility.

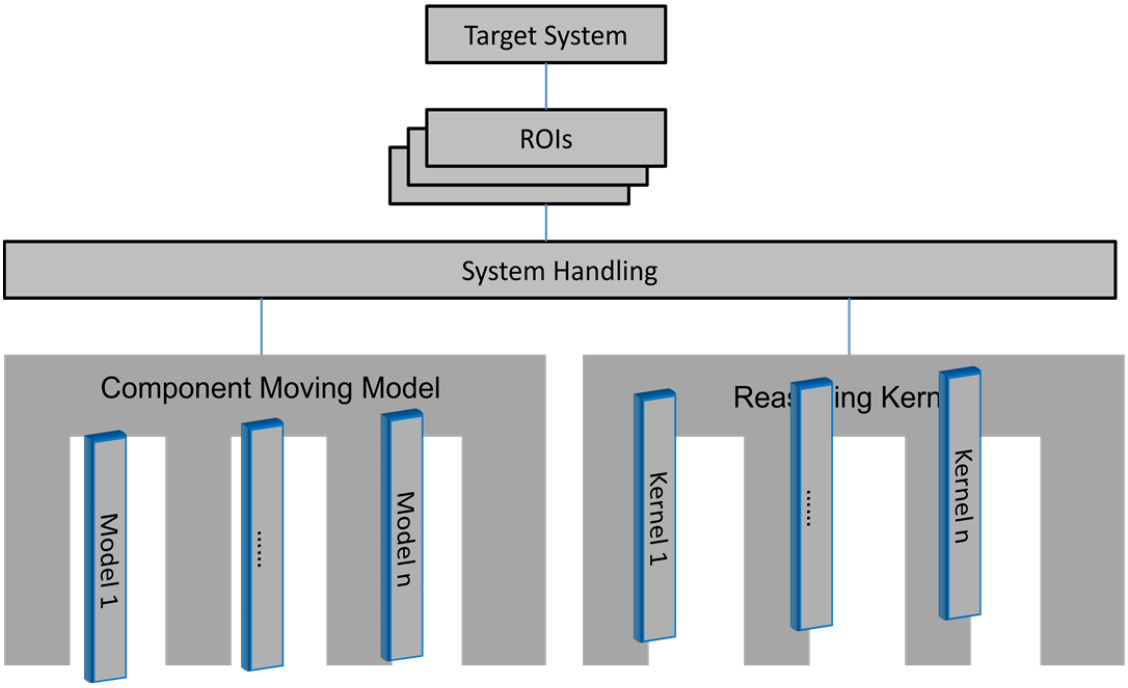


Figure 6-3 Suggested universal monitoring framework

6.3.3 Other possible improvements

In the research reported in this thesis, the process determined the reasoning kernel of the two types of abnormities (blocking and slowdown) in two steps. In the first step, the characteristics of an abnormity were examined mathematically with the assumption of constant inter-arrival time between components; in the following step, the results were forwarded to the reasoning scheme. Although the experimental results show good compliance in both cases of inter arrival time between components irrespective of whether they were constant or random, scientific proof of this deserves attention in future research.

In the proposed component moving model, the work-station processing times are assumed to be constant. However in some practical cases, it is possible for a work-station to encounter some deviations in its processing time. In addition, the proposed monitoring

method currently only focuses on the identification of abnormalities that have been occurring in the system. It will be more attractive if the preventive alerts can be issued during the monitoring, therefore related preventive maintenance or inspection can be conducted to prevent the occurrence of abnormality. Some intelligent technologies such as data mining and machine learning can be integrated into the proposed monitoring method to enhance the related capability in this field.

In addition, although the proposed method is capable of monitoring the two common abnormities, blocking and slowdown, in the production system, other key activities of the system can be reflected through ROI "Pulse" in future research. The relationship among these "Pulse" features and the system performance can then be addressed.

6.4 Possible applications of proposed methodology

Functions of the proposed model are not limited to production system monitoring. One possible application is illustrated in this section. The proposed model is also able to be applied to study the instant effectiveness of a production line by integrating the concept of Overall Equipment Effectiveness (OEE).

Competition from the global market drives most manufacturers to continuously pursue higher quality products at a lower cost. One feasible solution is to understand the efficiency of a current system so that appropriate actions can be undertaken to improve the productivity. Many productivity performance measurement approaches have been presented in the literature, and among them the Overall Equipment Effectiveness (OEE) is regarded as a fundamental tool for reflecting the efficiency of a manufacturing system

(Raja, Kannan, & Jeyabalan, 2012). OEE was initially proposed by Nakajima (1988) to analyze the efficiency of a single machine by three types of losses: availability losses due to machine breakdown and setup ($A_{eff} = T_U/T_L$), performance losses due to machine idling ($P_{eff} = (T_P/T_U) \times \left(R_{(avg)}^{(a)}/R_{(avg)}^{(th)}\right)$), and output losses due to product quality ($Q_{eff} = P_g/P_a$):

$$OEE = A_{eff} \times P_{eff} \times Q_{eff} = \frac{T_P}{T_L} \times \frac{R_{(avg)}^{(a)}}{R_{(avg)}^{(th)}} \times \frac{P_g}{P_a}$$

where T_U, T_L, and T_P represent the equipment uptime, loading time, and the production time respectively; $R_{(avg)}^{(a)}$ is the average actual output rate and $R_{(avg)}^{(th)}$ is the average theoretical output rate; P_g is the good product units within $T_L$and P_a is the actual product units within T_L.

Despite OEE having gained a lot of attention for its simple but comprehensive measurement, it is only suitable for the measurement of individual equipment items. Although several variations of OEE have been proposed to meet the requirements of a production line (**O**verall **L**ine **E**ffectiveness, **O**verall **E**quipment **E**ffectiveness of a **M**anufacturing **L**ine), or even a whole factory (**O**verall **F**actory **E**ffectiveness), the implementation is much more complex and strict hypotheses on relationships between internal equipment are posed simultaneously, which limits the application of these methods. Most importantly, the result of the measurement reflects only the average performance of the equipment/line/factory and does not indicate the instant performance fluctuation. Embedding the proposed model in the concept of OEE can address this gap.

Regarding a single ROI as an individual piece of equipment, the predicted throughput time of a work-piece is T_P while the actual throughput time is T_L. The reciprocal of the predicted work-piece's inter-leaving time from the ROI is the theoretical output rate $R^{(th)}$ and the reciprocal of the actual ILT is the actual output rate $R^{(a)}$. Since non-qualified work-pieces will be scrapped from the ROI, the predicted instant WIP and the actual instant WIP can be regarded as P_g and P_a respectively. With this evolutionary OEE method, the instant performance of a production line (ROI) can be measured online, getting rid of the hypothesis on internal relationships.

Chapter Seven - Conclusion

The contribution of this research is the development of an innovative technique to monitor the healthiness of a production line by observing the "pulse chain". This has been done by modelling a manufacturing system as the integration of Regions of Interest (ROIs). For each ROI, a pair of counter devices with a time-stamp function is installed at both the Inlet and the Outlet to collect the meta-information of components, and the dynamic behaviour of the system is abstracted by monitoring the components/work-pieces flow characteristics within these ROIs. The proposed technique was validated through simulation experiments and the results proved encouraging.

Several features were extracted to constitute the "pulses" of an ROI, including Regional Inconsistency (RI), Inter-component Arrival Time (IAT), Inter-component Leaving Time (ILT), and Instant Work-In-Progress (IWIP). Reasoning schemes for two types of common abnormalities (blocking and slowdown) in an ROI were established. First, the line blocking symptom was examined and the mathematical model for the determination of the Balanced Blocking Point (BBP) was introduced. Then, by tracing the changes of IAT, ILT, and IWIP, the line blocking could be detected and subsequently located. In terms of operations, one should keep watching whether or not there is a ΔT_{io} case; once a change has been confirmed, the "BP Locations vs ΔT_{io}" graph can be used to narrow down the scope of the blocking point location in the production line. The second abnormality covered in this research is about the slowdown in a production line. Based on the developed **F**our-**L**ayer **F**ilter **A**lgorithm (**FLFA**) and the formulated mathematical model, the slowdown symptoms, including slight machine slowdown, slight transfer

slowdown, serious machine slowdown and serious transfer slowdown can be signified accordingly.

In terms of the monitoring system design aspect, the **Three-Step ROI Segmentation** technique was developed to guide the design, subjected to a specified maximum response time. Each segmented ROI constitutes the basic observation unit of the monitoring system and it is anticipated that, based on the information from these ROIs, some typical production problems can be identified.

References

Abdel-Hamid K.T. (1989). The dynamics of software project staffing: A system dynamics based simulation approach. IEEE Transactions on Software Engineering, 15 (2), 109-119.

Al-Ahmari M.A.A. (2007). Evaluation of CIM technologies in saudi industries using AHP. International Journal of Advanced Manufacturing Technology, 34 (7), 736-747.

Amasaka Kakuro. (2009). TPS-QAS, new production quality management model: key to New JIT - Toyota's global production strategy. International Journal of Manufacturing Technology and Management, 18 (4), 409-426.

Angerhofer B.J. (2000). System dynamics modelling in supply chain management: research review. Proceedings of the 2000 Winter Simulation Conference, 342-351, Piscataway, NJ.

Bai J., Wu D., & Zhang J. (1994). Computer simulation of pulse wave. Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings, 11, p118/1 - 118/2. Beijing, China.

Baines S.T. (1994). Modelling in the evaluation of a manufacturing strategy.

Baines S.T., & Harrison K.D. (1999). An opportunity for system dynamics in manufacturing system modelling. Production Planning & Control: The Management of Operations, 10 (6), 542-552.

Benton W., & Shin H. (1998). Manufacturing planning and control: the evoluation of MRP and JIT Integration. European Journal of Operational Research, 1 (10), 411-440.

Blanchard S. Benjamin. (2008). System engineering management. Hoboken, New Jersey: John Wiley & Sons, Inc.

Brailsford C.S., & Hilton A.N. (2000). A comparison of discrete event simulation and system dynamics for modelling healthcare systems. Proceedings of ORAHS (p18-39). Glasgow Caledonian University.

Byrne J.S., & Roberts L. (1994). Efficient parts supply: influence of information flows. 1994 International Ssytem Dynamics Conference (pp. 1-19). University of Stirling, UK: Productin and Operations Management.

Chang C.C., & Yang T.Y. (2004). A correlation between pulse diagnosis of human body and health monotoring of structures. Earthquake Engineering and Engineering Vibration, 3 (1), 117-125.

Dey S., & Stori A.J. (2005). A bayesian network approach to root cause diagnosis of process variations. International Journal of Machine Tools and Manufacture, 45 (1), 75-91.

Dimla E. (2000). Sensor signals for tool-wear monitoring in metal cutting operations: A review of methods. International Journal of Machine Tools and Manufacture, 40 (8), 1073-1098.

Dimla E.D., Lister E.P., & Leighton J.N. (1997). Neural network solution to the tool condition monitoring problem in metal cutting - a critical review of methods. International Journal of Machine Tools and Manufacture, 37 (9), 1219-1241.

Dombrowski U., Crespo I., & Zahn T. (2010). Adattive configuration of a lean production system in small and medium-sized enterprises. Production Management, 4 (4), 341-348.

Dotoli M., Fanti P.M., & Mangini M.A. (2008). Fault monitoring of discrete event systems by first order hybrid petri nets. In Workshop on Petri Nets and Agile Manufacturing, a satellite event of the 29th Int. Conf. on Application and Theory of Petri Nets and Other Models of Concurrency, (pp. 1-7). Xi'an, China.

Du X.R., Elbestawi A.M., & Li S. (1992). Tool condition monitoring in turning using fuzzy set theory. International Journal of Machine Tools and Manufacturing, 32 (6), 781-796.

Edghill S.J., & Towill R.D. (1989). The use of system dynamics in manufacturing systems engineering. Transactions of the Institute of Measurement and Control, 11 (4), 208-216.

Edmunds A., & Morris A. (2000). The problem of information overload in business organisations: A review of the literature. International Journal of Information Management, 20, 17-28.

Forrester Wright Jay. (1961). Industrial dynamics. Cambridge, Mass.: M.I.T. Press.

Frank M.P., Ding X.S., & Marcu T. (2000). Model-based fault diagnosis in technical processes. Transactions of the Institution of Measurement and Control, 22 (1), 57-101.

George H.L., Kelly R.G., Jeff J.A., & Thoney K. (2011). Adapting lean manufacturing principles to the textile industry. Production Planning & Control: The Management of Operations, 22 (3), 237-247.

Gourgand M., Lacomme P., & Traore K.M. (2003). Design of a monitoring environment for manufacturing systems management and optimization. International Journal of Computer Integrated Manufacturing, 16 (1), 61-80.

Guh S.R. (2010). Simultaneous process mean and variance monitoring using artificial neural networks. Computer and Industrial Engineering, 58 (4), 739-753.

Gunasekaran A., Marri B.H., & Lee B. (2000). Design and Implementation of Computer Integrated Manufacturing in Small and Medium-Sized Enterprises: A Case Study. International Journal of Advanced Manufacturing Technology, 16, 46-54.

Hansen E.J., & Bie P. (1987). Distribution of body fluids, plasma protein, and sodium in dogs: a system dynamic model. System Dynamics Review, 3 (2), 116-135.

Hitomi K. (1991). Strategic integrated manufacturing systems: The concept and structures. International Journal of Production Economics, 25 (1), 5-12.

Homer B.J., & Clair L.S.C. (1991). A model of HIV transmission through needle sharing. Interfaces, 21 (3), 26-29.

Hunter L.Steve. (2008). The Toyota Production System applied to the upholstery furniture manufacturing industry. Materials and Manufacturing Processes, 23 (7), 629-634.

INCOSE. (2004). Systems Engineering Handbook, INCOSE-TP-2003-016-02, Version 2a.

Jardim-Goncalves R., Silva H., Vital M., Sousa P., Seiger-Garcao A., & Pamies-Teixeira J. (1997). Implementation of computer integrated manufacturing systems using SIP: Cim case studies using a STEP approach. International Journal of Computer Integrated Manufacturing, 10 (1-4), 172-180.

Kahraman Cengiz, & Tuysuz Fatih. (2010). Manufacturing system modelling using Petri nets. Production Engineering and Management, 252, 95-124.

Kalir A., & Arzi Y. (1997). Automated production line design with flexible unreliable machines for profit maximization. International Journal of Production Research, 35 (6), 1651-1664.

Koren Yoram. (1983). Computer Control of Manufacturing Systems. New York, USA: McGraw-Hill, Inc.

Koriem M.S. (1999). R-nets for the performance evaluation of hard real-time systems. Journal of Systems and Software, 46 (1), 41-58.

Kramer A.M., & MahS.H.R. (1993). Model-based monitoring. Proceedings of Second International Conference on "foundations of Computer-Aided Process Operations", (pp. 45-68). Austin.

Kuo C-H, & Huang H-P. (2000). Failure modelling and process monitoring for flexible manufacturing systems using colored timed Petri nets. IEEE Transactions on Robotics and Automation, 16 (3), 301-312.

Lara A.M., & Nof Y.S. (2003). Computer-supported conflict resolution for collaborative facility designers. International Journal of Productiton Research, 41 (2), 207-233.

Lee- H.B., & Jo- J.H. (2007). The mutation of the Toyota Production System: adapting the TPS at Hyundai Motor Company. International Journal of Production Research, 45 (16), 3665-3679.

Lee Hsing-Lin, Suzuki S., Adachi Y., & Umeno M. (1993). Fuzzy theory in traditional Chinese pulse diagnosis. Proceeding of International Joint Conference on Neural Networks, (774-777). Nagoya, Japan.

Lee Jay. (1998). Teleservice engineering in manufacturing: challenges and opportunities. International Journal of Machine Tools and Manufacture, 38 (8), 901-910.

Lewis W.R., & Ransing S.R. (1997). A semantically constrained bayesian network for manufacturing diagnosis. International Journal of Production Research, 35 (8), 2171-2187.

Lewoc B.J., Izworski A., Skowronski S., Kieleczawa A., & Kopacek P. (2011). Design and development of Computer Integrated Manufacturing and management systems for manufacturing enterprises. International Journal of Latest Trends in Computing, 2 (4), 508-515.

Liker Jeffrey. (2003). The Toyota Way: 14 Management Principles from the World's Greatest Manufacturer. New York: McGraw-Hill.

Lin Chang-Pin, & Jeng MuDer. (2006). An expanded SEMATECH CIM framework for heterogeneous applications integration. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 36 (1), 76-90.

Ly F., Toguyeni K.A.A., & Craye E. (2000). Indirect predictive monitoring in flexible manufacturing systems. Robotics and Computer Integrated Manufacturing, 16 (5), 321-338.

Macarthy L.B., & Wasusri Thananya. (2002). A review of non-standard applications of statistical process control (SPC) charts. International Journal of Quality & Reliability Management, 295-320.

Masood, T., & Khan, I. (2004). Productivity improvement through computer-integrated manufacturing in post WTO scenario. *Proceedings of the National Conference on Energy Technologies*, (pp. 171-177).

Mcgaughey E.R., & Roach D. (1997). Obstacles to computer integrated manufacturing success: A study of practitioner perceptions. International Journal of Computer Integrated Manufacturing, 10 (1-4), 256-265.

Mcgaughey E.R., & Snyder A.Charles. (1994). The obstacles to successful CIM. International Journal of Production Economics, 37, 247-258.

Mcgehee John, Hebley John, & MahaffeyJack. (1994). The MMST Computer-Integrated Manufacturing System Framework. IEEE Transactions on Semiconductor Manufacturing, 7 (2), 107-116.

Merdan M., Vallee M., Lepuschitz W., & Zoitl A. (2011). Monitoring and diagnostics of industrial systems using automation agents. International Journal of Production Research, 49 (5), 1497-1509.

Nagalingam V.S., & Lin C. I.G. (2008). CIM-still the solution for manufacturing industry. Robotics and Computer-Integrated Manufacturing, 24 (3), 332-344.

Nakajima S. (1988). Introduction to TPM: Total Productive Maintenance. Cambridge, MA: Productivity Press.

Nakamura M., Sakakibara S., & Schroeder R. (1996). Japanese manufacturing method at US manufacturing plants: empirical evidence. The Canadian Journal of Economics, S468-S474.

Ohno Taiichi. (1988). Toyota production system: beyond large-scale production. Cambridge: Productivity Press.

Oztemel Ercan, & Tekez Kurt Esra. (2009). A generic framework of a reserence model for intelligent integrated manufacturing systems (REMIMS). Engineering Applications of Artificial Intelligence, 22 (6), 855-864.

Parnaby J. (1979). Concept of a manufacturing system. International Journal of Production Research, 17 (2), 123-135.

Piramuthu S., Shaw M., & Fulkerson B. (2000). Information-based dynamic manufacturing system scheduling. International Journal of Flexible Manufacturing Systems, 12 (2-3), 219-234.

Qian Chen, Chan YuenChing, Tang ShingChi, & Yung LeungKai. (2011). System monitoring through element flow reasoning. Robotics and Computer-Integrated Manufacturing, 27 (1), 221-233.

Raja N.P., Kannan M.S., & Jeyabalan V. (2012). Overall line effectiveness - a performance evaluation index of a manufacturing system. International Journal of Productivity and Quality Management, 5 (1), 38-59.

Rehg A.James, & Kraebber W.Henry. (2005). Computer-Integrated Manufacturing. New Jersey: Pearson Education.

Richardson P.G., & Pugh L.A. (1981). Introduction to system dynamics modelling with DYNAMO. Cambridge, Mass.: M.I.T. Press.

Rodrigues L.R.L., Dharmaraj N., & RaoShrinivasa B.R. (2006). System dynamics approach for change management in new product development. Management Research News, 29 (8), 512-523.

Sandoval Victor. (1994). Computer Integrated Manufacturing (CIM) in Japan. Netherlands: Elsevier Science B. V.

Sick B. (2002). On-line and indirect tool wear monitoring in turning with artificial neural networks: A review of more than a decade of research. Mechanical Systems and Signal Processing, 16 (4), 487-546.

Telmoudi J.A., Nabli L., & Bourjault A. (2008). Symptoms detection in FMS through performance indicators follow-up: a quality-flow approach. 2008 International Conference on Condition Monitoring and Diagnosis, (pp. 803-807). Beijing, China.

Toguyeni A., & Korbaa O. (2005). Indirect monitoring of the failures of a flexible manufacturing system under cyclic scheduling. Rototics and Computer-Integrated Manufacturing, 21 (1), 1-10.

Tseng C.H., Ip H.W., & Ng C.K. (1999). A model for an integrated manufacturing system implementation in China: a case study. Journal of Engineering and Technology Management, 16 (1), 83-101.

Venkatasubramanian V., Rengaswamy R., & Kavuri N.S. (2003). A review of process fault detection and diagnosis part 2: Qualitative models and search strategies. Computers and Chemical Engineering, 27 (3), 313-326.

Womack P.James, & Jones T.Daniel. (2003). Lean thinking: banish waste and create wealth in your corporation. New York: New York : London : Free.

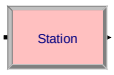
Yurdakul Mustafa. (2004). Selection of computer-integrated manufacturing technologies using a combined analytic hierarchy process and goal programming model. Robitics and Computer-Integrated Manufacturing, 20 (4), 329-340.

Zaremba B.Marek, & Prasad Biren. (1994). Modern Manufacturing: Information Control and Technology. London: Springer-Verlag London Limited.

Zhu P.K., Wong S.Y., & Hong S.G. (2009). Multi-category micro-milling tool wear monitoring with continuous hidden markov models. Mechanical Systems and Signal Processing, 23 (2), 547-560.

Appendix A - Experiments Setup in Arena

The process of experiments in Arena was classified into two steps. First the target production line was first constructed by the preset modules provided in Arena. In our experiments the major employed modules were listed as follows.



Module “Station”, corresponding to the physical Work-stations

Module “Process”, corresponding to the processing of Work-stations

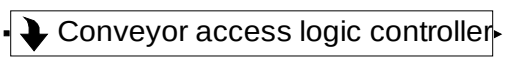
Module “Convey”, corresponding to the beginning of conveyor segment

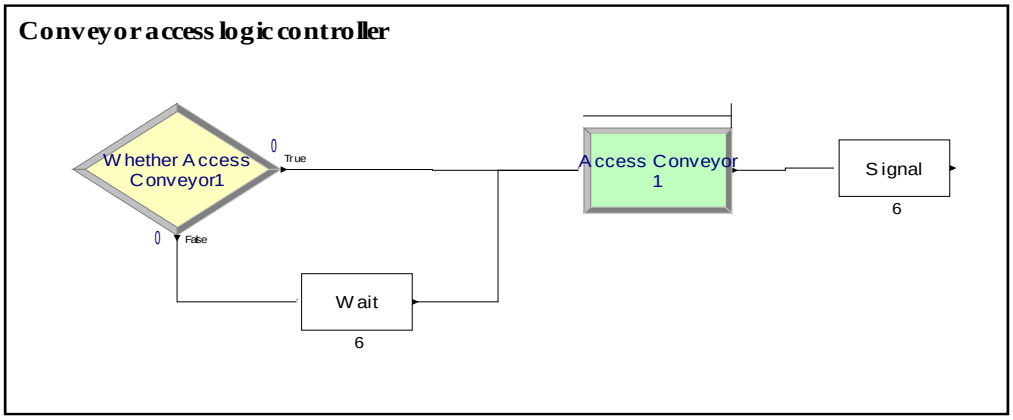
Module “Exit”, corresponding to the end of conveyor segment

Module “Create”, starting point for components in the model

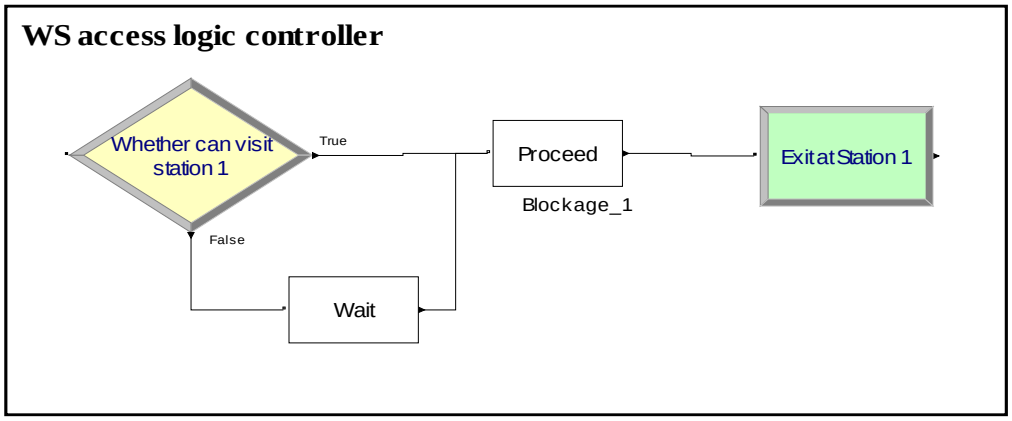
Module “Dispose”, ending point for components in the model

Besides these preset modules, the following three self-built modules, so-called sub-models in Arena, were also developed to conduct the logic control. Conveyor access logic controller and WS access logic controller were developed to control whether the component can access the conveyor/work-station or not while the function of inlet time-stamp counter and outlet time-stamp counter, just as their names suggest, were just to record the index and the clock time when components arrives at the inlet and leaves from the outlet.

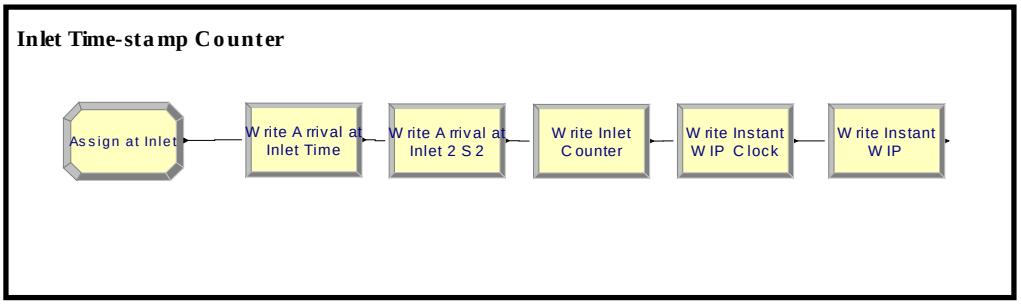


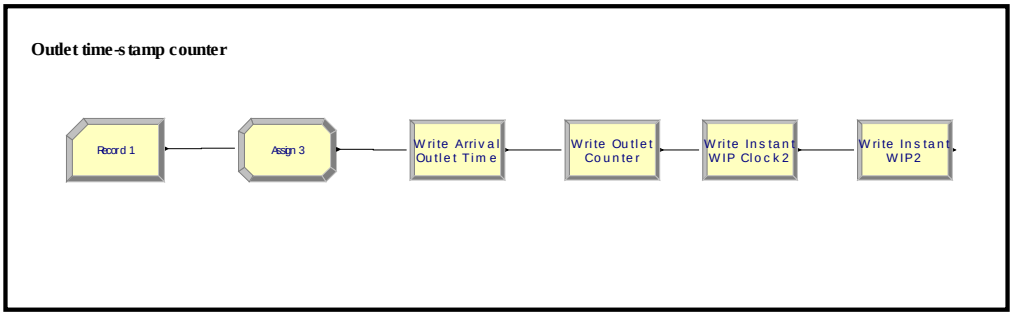


WS access logic controller



Inlet Time-stamp Counter





With these preset modules and self-built modules, the production line can be established and various scenarios of defects within the line can be simulated.

Experiment One: Blocking cases within Five-Work-station single ROI

The work-stations layout of the ROI is shown in Figure A-1. In steady state, components arrive at the inlet and leave from the outlet with the logic control flow in Figure A-2.

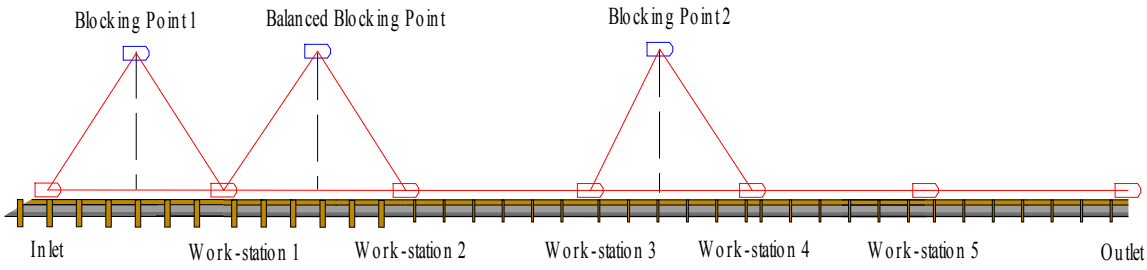


Figure A - 1 Work-station layout of the Five-Work-station single ROI

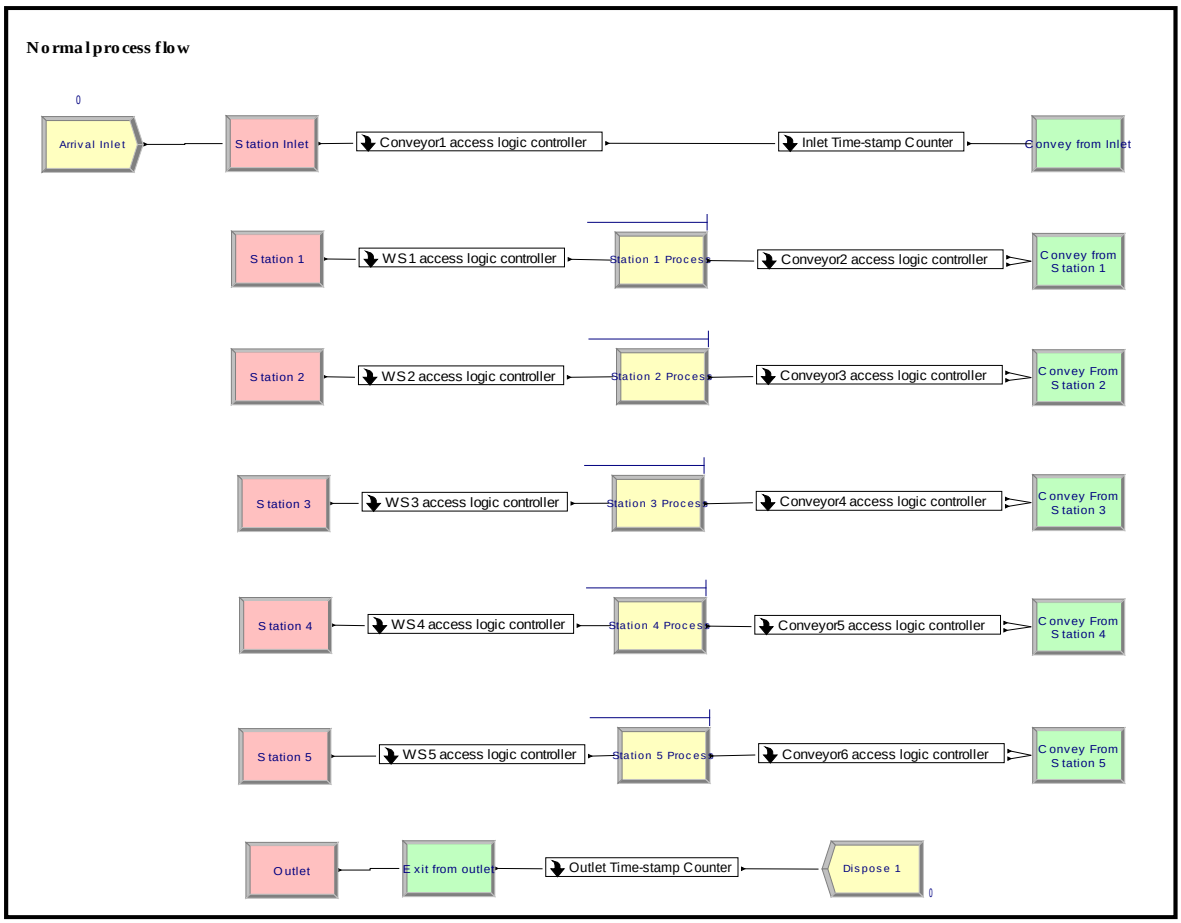
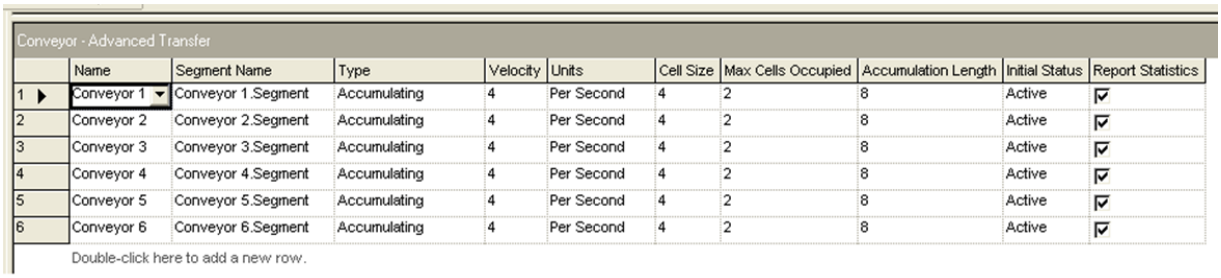


Figure A - 2 Logic control flow of the Five-Work-station ROI

The transfer time plays an essential role in our proposed module and it was implemented by means of "Conveyor" module in Arena to control the components movement between work-stations, as Figure A-3 shows. Taking Conveyor1 for example, the field "Segment name" associates Conveyor1 with the specified segment set "Conveyor1.Segment" in Figure A-4, which contains two segments, 60 from "Station Inlet" to "Station Blocking" and 60 from "Station Blocking" to "Station 1". With the "Velocity" of 4 per second, the ideal transfer time from the inlet to the 1^{st} work-station was set as 30 time units and location of the "Station Blocking" was at 15 time units after the inlet, as the Table 5-2 showed. The conveyor was set as "Accumulating" type and the

fields of “Cell Size”, “Max Cells Occupied”, and “Accumulation Length” were set to 4, 2, and 8 respectively to guarantee the minimum gap between components T_g to be 2 (Arena Online Help).



Conveyor - Advanced Transfer

	Name	Segment Name	Type	Velocity	Units	Cell Size	Max Cells Occupied	Accumulation Length	Initial Status	Report Statistics
1	Conveyor 1	Conveyor 1.Segment	Accumulating	4	Per Second	4	2	8	Active	☑
2	Conveyor 2	Conveyor 2.Segment	Accumulating	4	Per Second	4	2	8	Active	☑
3	Conveyor 3	Conveyor 3.Segment	Accumulating	4	Per Second	4	2	8	Active	☑
4	Conveyor 4	Conveyor 4.Segment	Accumulating	4	Per Second	4	2	8	Active	☑
5	Conveyor 5	Conveyor 5.Segment	Accumulating	4	Per Second	4	2	8	Active	☑
6	Conveyor 6	Conveyor 6.Segment	Accumulating	4	Per Second	4	2	8	Active	☑

Double-click here to add a new row.

Figure A - 3 Settings of “Conveyor” module

The established virtual production line should first be validated before being employed to conduct the experiments. By adjusting components arrival patterns to different distributions in “Create” module, we compared the components leaving times in Arena with the outcomes obtained through the proposed prediction method and the results coincided perfectly.

To simulate the blocking cases, blocking points were simulated as “Station” with “Delay” function. In normal state, the delay time was “0” and the station could be ignored actually; when blocking occurred the delay duration was set as “100000” time units to block the components. Logic control process of blocking stations was as Figure A-4 and the route control of components flowing through “Blocking station” was completed through the settings in “Convey” and “Segment” modules. Figure A-5(a) and A-5(b) show the settings of blocking before and after BBP respectively.

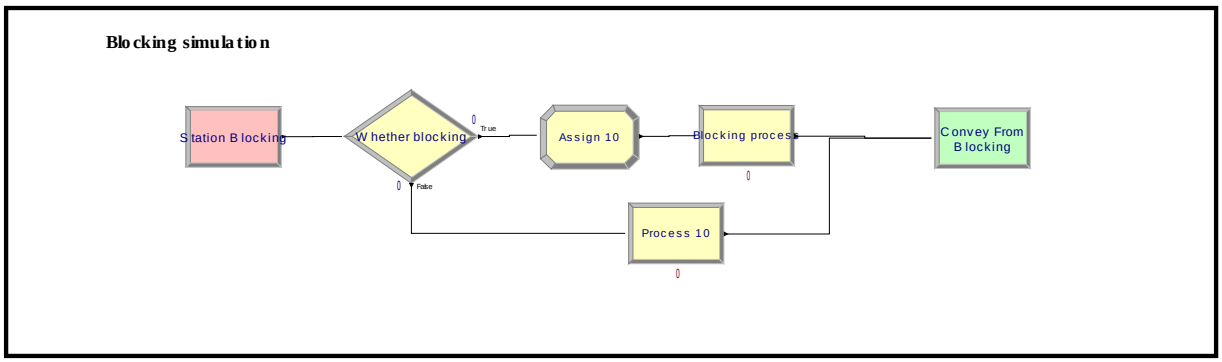
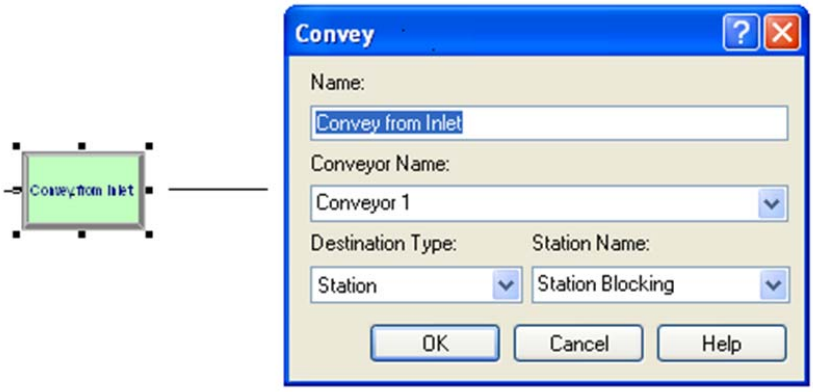
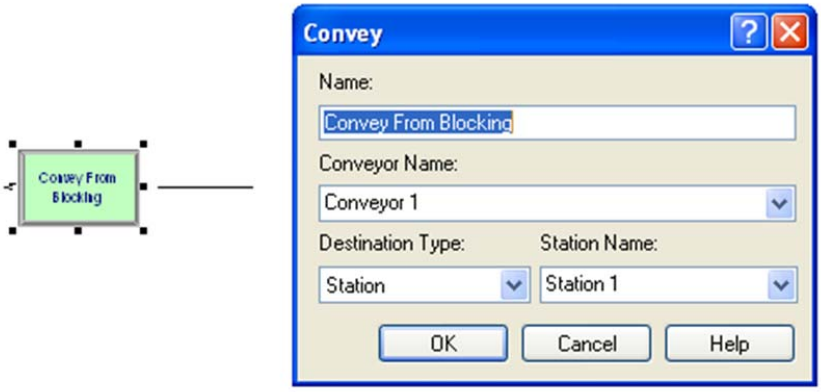


Figure A - 4 Logic control process of the blocking station





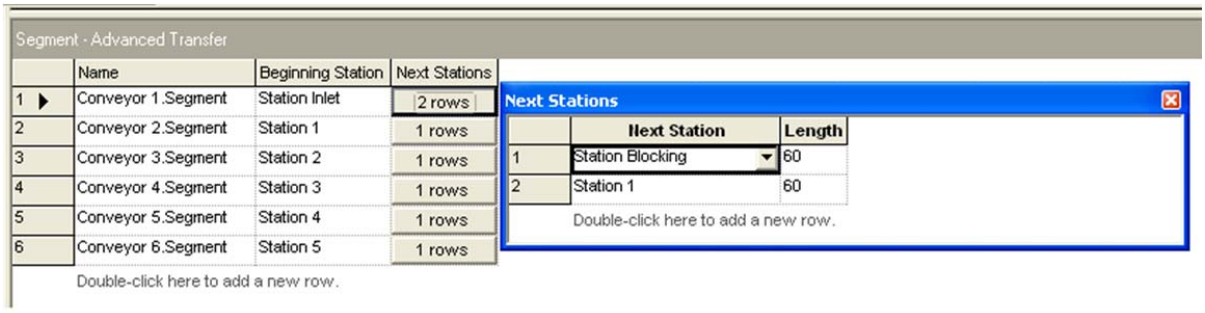
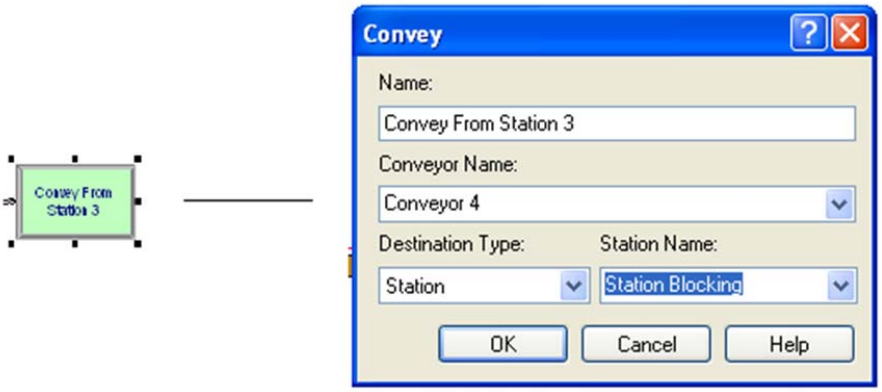
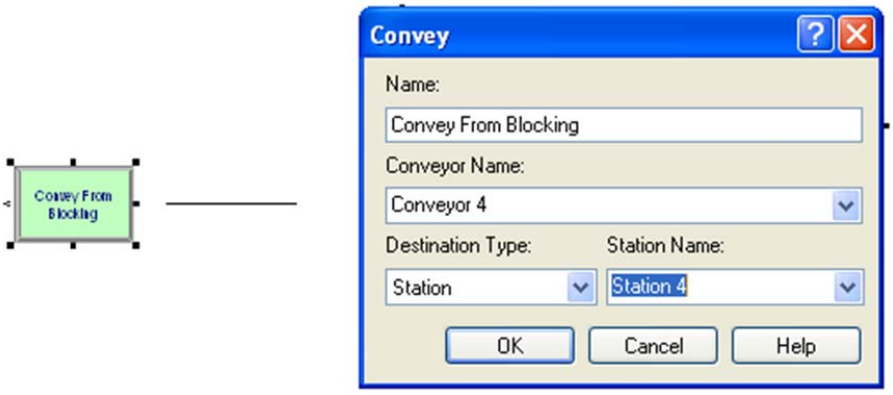


Figure A - 5(a) Settings in “Convey” and “Segment” modules (blocking before BBP)





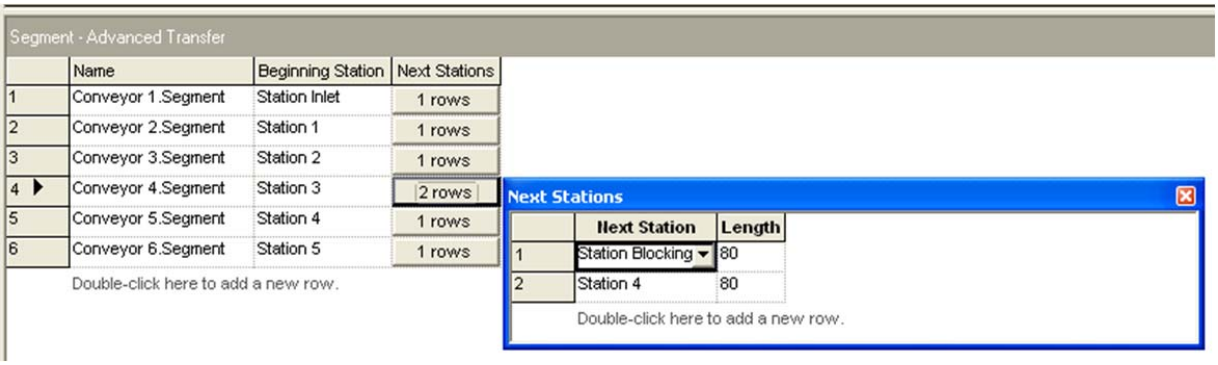


Figure A - 5(b) Settings in "Convey" and "Segment" modules (blocking after BBP)

Experiment Two: Slowdown cases within Five-Work-station single ROI

The process layout of slowdown cases was the same with that of blocking cases and the occurrence of slowdown was implemented through the alteration of station process time and the velocity of conveyors. Figure A-6(a) shows the logic control of Work-station slowdown cases. The processing time of stations 4 was set as a variable of "M4_PT". In normal situation, in normal status the variable was set a value in "Assign 7" module; when slowdown occurred, however, the variable was set to another in "Assign 8" module. For transfer slowdowns, the velocity of the conveyor was modified to an appropriate value in "Assign 9" module according to the scenarios (see Figure A-6(b)).

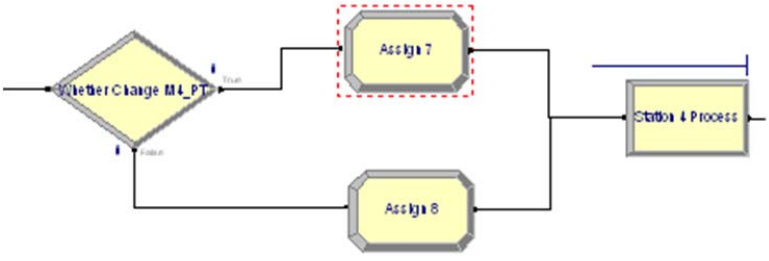


Figure A - 6(a) Settings of Work-station slowdown cases

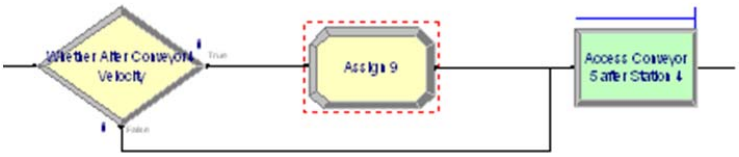


Figure A - 6(b) Settings of Transfer slowdown cases

Experiment Three: ROI segmentation of a 15 work-station production line

To validate the effectiveness of the Three-Step ROI Segmentation technique, a virtual production line containing 15 work-stations were constructed in Arena® based on the parameters settings in Table 5-4. Corresponding to the calculation result in Figure 5-9, the whole line was segmented into five ROIs (see Figure A-7). To explicitly distinguish these five ROIs, five separate lines were listed in Figure A-7 to represent these ROIs respectively. For each pair of two adjacent ROIs, the outlet of the preceding ROI and the inlet of the following ROI shared the same counter. For instant, "Outlet1" and "Inlet2" represented the same physical counter although they appeared at two separate locations in Figure A-7. Two outlet counters were setup at the exit of work-station 11 and work-station 15 separately, which also served as the inlet of ROI4 and ROI5. Logic controls of these five ROIs were shown in Figure A-8.

Blocking and slowdown scenarios were conducted at different locations along the production line and the method was similar as the experiments in Experiment One and Two. The response time were collected accordingly. For example Figure A-9 shows the blocking locations within ROI1; from where the blocking was set up to occur at these 7 locations separately. Slowdown cases were conducted on all work-stations and branches respectively. Similar experiments were conducted on ROI2, ROI3, and ROI4.

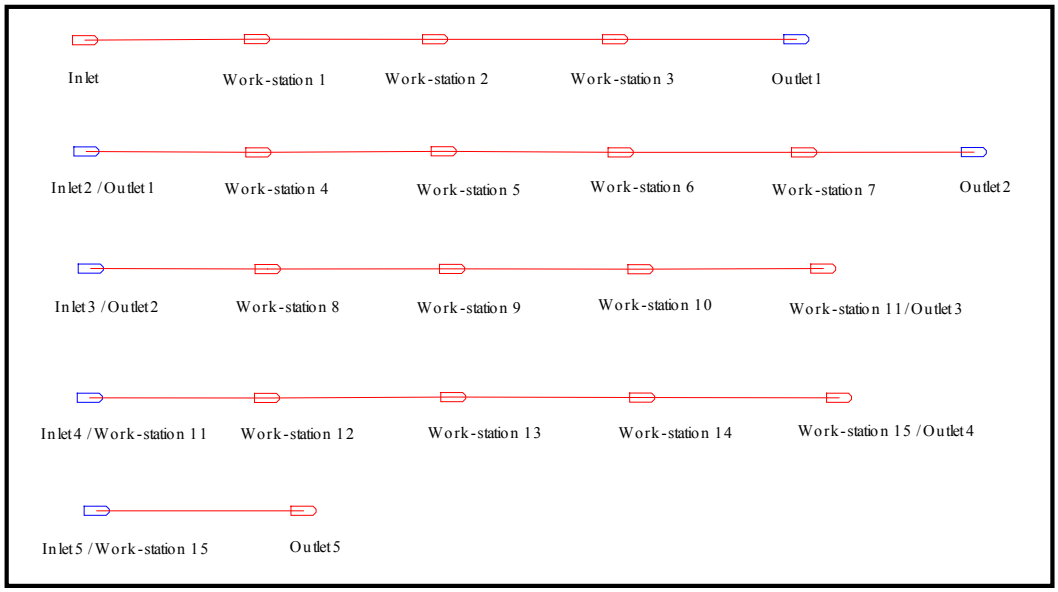


Figure A - 7 Facility layout of ROI Segmentation result

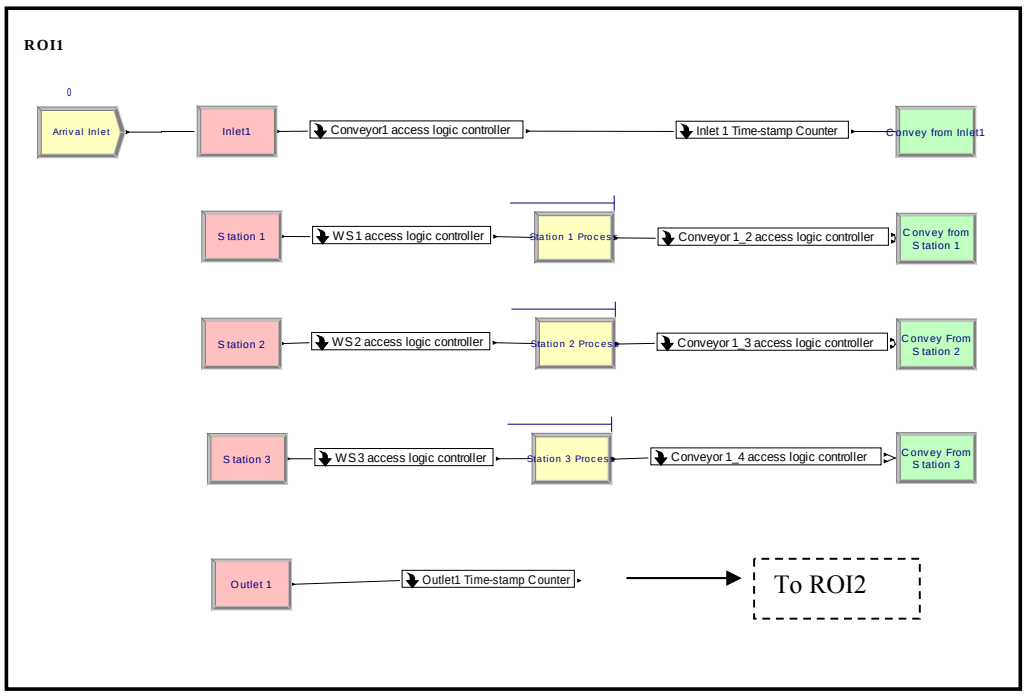


Figure A-8(a) Logic control process of the segmented ROIs

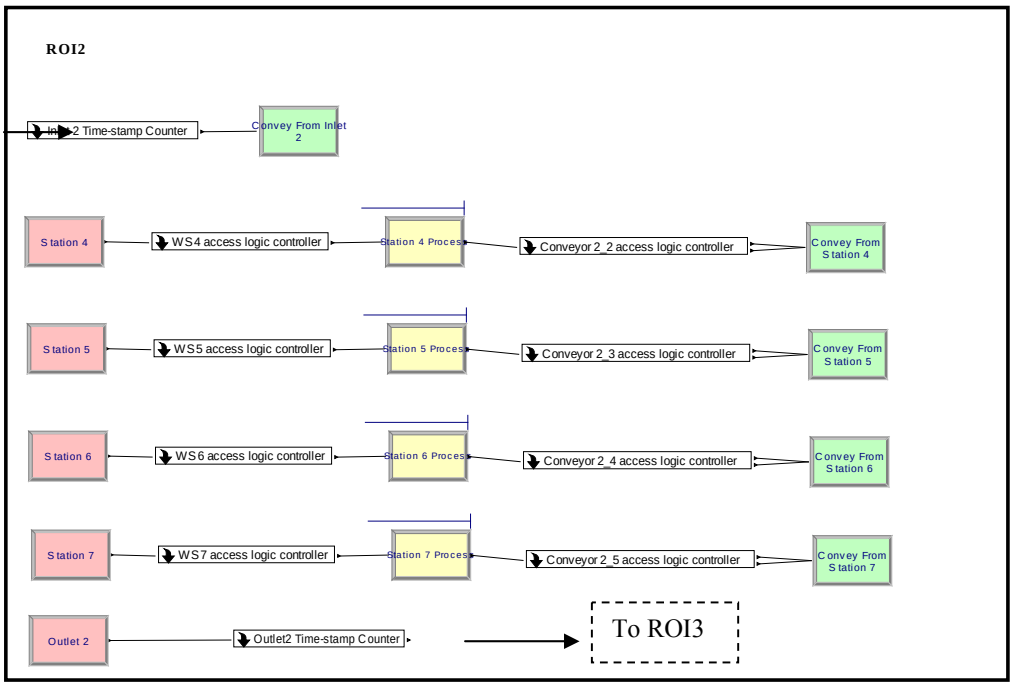


Figure A - 8(b) Logic control process of the segmented ROIs

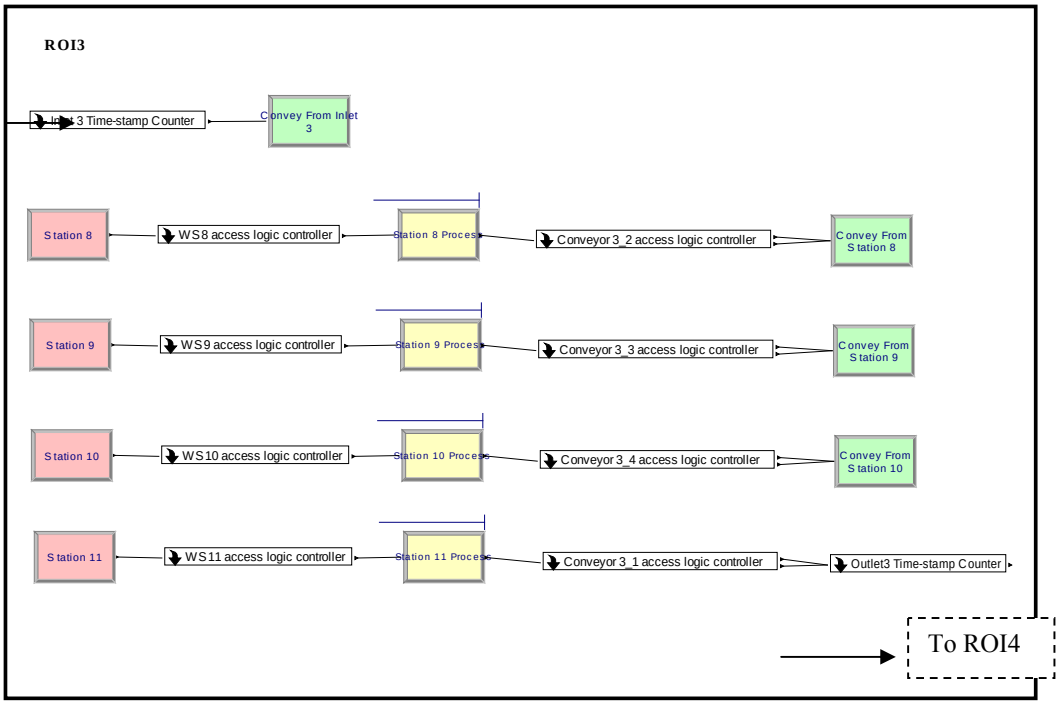


Figure A - 8(c) Logic control process of the segmented ROIs

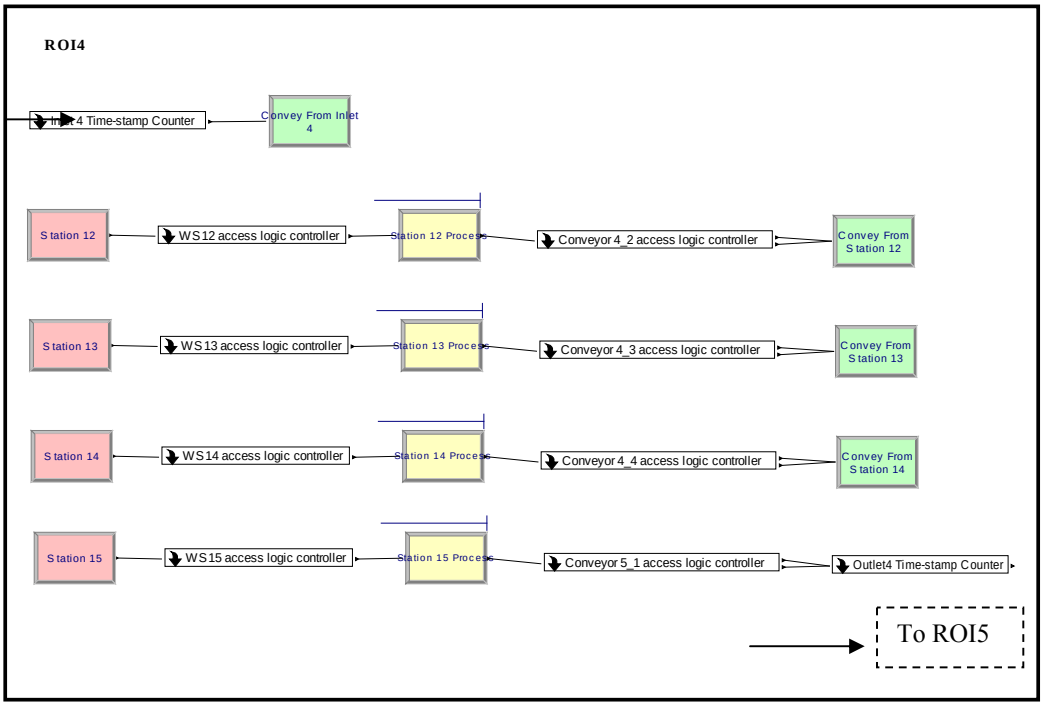


Figure A - 8(d) Logic control process of the segmented ROIs

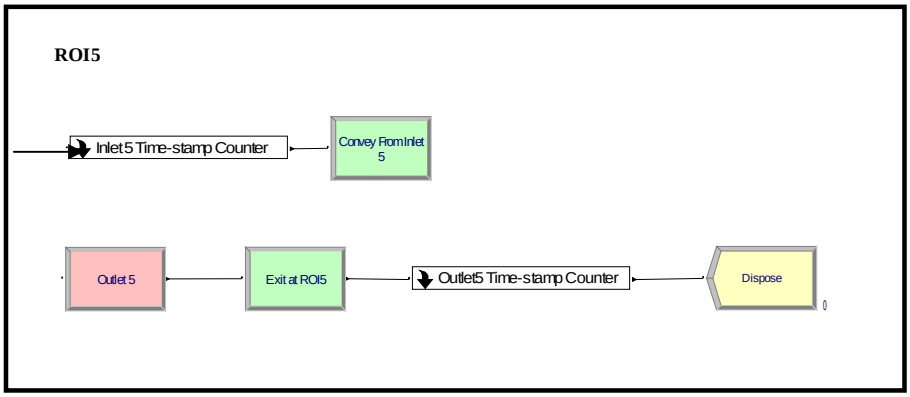


Figure A - 8(e) Logic control process of the segmented ROIs

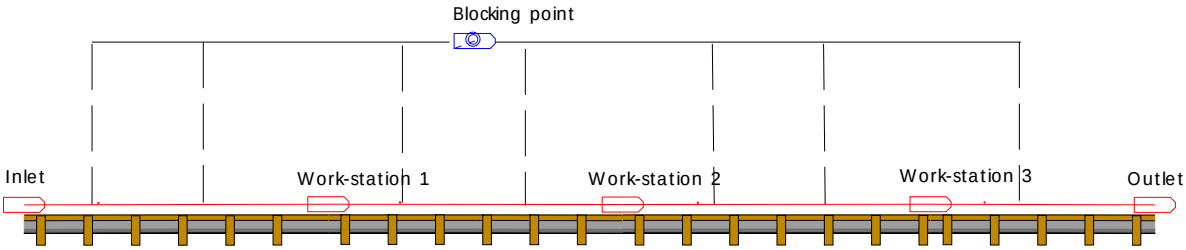


Figure A - 9 Blocking point setup within ROI1

Appendix B-1 - Data of blocking case before BBP (random inter-arrival time)

Index (β)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$t_{i,\beta}$	0	2	5	7	9	11	13	26	31	43	54	56	65	84	86	89	94	96	98	100	120	132	145	181	192
IAT	-	2	3	2	2	2	2	13	5	12	11	2	9	19	2	3	5	2	2	2	20	12	13	36	11
$t_{o,\beta}$	185	190	195	200	205	210	215	220	225	230	239	244	250	269	274	279	284	289	294	299	305	317	330	366	377
ILT	-	5	5	5	5	5	5	5	5	5	9	5	6	19	5	5	5	5	5	5	6	12	13	36	11
Index (β)	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$t_{i,\beta}$	200	208	210	214	216	218	225	228	236	242	244	255	264	266	287	297	304	308	311	313	315	332	345	347	352
IAT	8	8	2	4	2	2	7	3	8	6	2	11	9	2	21	10	7	4	3	2	2	17	13	2	5
$t_{o,\beta}$	385	393	398	403	408	413	418	423	428	433	438	443	449	454	472	482	489	494	499	504	509	517	530	535	-
ILT	8	8	5	5	5	5	5	5	5	5	5	5	6	5	18	10	7	5	5	5	5	8	13	5	-
Index (β)	51	52	53	54	55	56	57																		
$t_{i,\beta}$	366	370	372	380	397	402	407																		
IAT	14	4	2	8	17	5	5																		
$t_{o,\beta}$																									
ILT																									

Appendix B-2 - Data of blocking case after BBP (random inter-arrival time)

Index (β)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$t_{i,\beta}$	0	2	4	6	15	17	22	26	29	31	38	43	46	48	50	56	70	72	74	92	94	101	103	105	107
IAT	-	2	2	2	9	2	5	4	3	2	7	5	3	2	2	6	14	2	2	18	2	7	2	2	2
$t_{o,\beta}$	185	190	195	200	205	210	215	220	225	230	235	240	245	250	255	260	265	270	275	280	285	290	295	300	305
ILT	-	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Index (β)	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$t_{i,\beta}$	111	113	116	118	123	132	145	158	169	177	181	183	185	197	220	224	229	236	238	253	256	261	269	276	278
IAT	4	2	3	2	5	9	13	13	11	8	4	2	2	12	23	4	5	7	2	15	3	5	8	7	2
$t_{o,\beta}$	310	315	320	325	330	335	340	345	354	362	367	372	377	382	405	410	415	421	426	438	443	448	454	461	
ILT	5	5	5	5	5	5	5	5	9	8	5	5	5	5	23	5	5	6	5	12	5	5	6	7	
Index (β)	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
$t_{i,\beta}$	282	284	291	293	295	298	300	302	307	316	322	325	338	340	358	361	370	378	380	391	398	404	411	426	436
IAT	4	2	7	2	2	3	2	2	5	9	6	3	13	2	18	3	9	8	2	11	7	6	7	15	10
$t_{o,\beta}$																									
ILT																									
Index (β)	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	
$t_{i,\beta}$	444	450	456	460	464	477	488	492	496	498	504	506	510	517	529	531	533	545	547	549	559	568	585	587	
IAT	8	6	6	4	4	13	11	4	4	2	6	2	4	7	12	2	2	12	2	2	10	9	17	2	
$t_{o,\beta}$																									
ILT																									

Appendix C-1 - Data of slowdown with constant inter-arrival time (Slight Machine Slowdown)

Index (β)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$t_{o,\beta}$	185	191	197	203	209	215	221	227	233	239	245	251	257	263	269	275	281	287	293	299	305	311	317	323	329
IAT	-	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
ILT	-	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
RI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ΔILT	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Index (β)	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$t_{o,\beta}$	335	341	347	353	359	365	371	377	383	389	395	401	407	413	419	425	431	437	443	449	455	461	467	473	483
IAT	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
ILT	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	10
RI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
ΔILT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
Index (β)	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
$t_{o,\beta}$	489	495	501	507	513	519	525	531	537	543	549	555	561	567	573	579	585	591	597	603	609	615	621	627	633
IAT	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
ILT	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
RI	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
ΔILT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Index (β)	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
$t_{o,\beta}$	639	645	651	657	663	669	675	681	687	693	699	705	711	717	723	729	735	741	747	753	759	765	771	777	783
IAT	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
ILT	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
RI	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
ΔILT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Appendix C-2 - Data of slowdown with constant inter-arrival time (Serious Machine Slowdown)

Index (β)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$t_{o,\beta}$	185	191	197	203	209	215	221	227	233	239	245	251	257	263	269	275	281	287	293	299	305	311	317	323	329
IAT	-	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
ILT	-	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
RI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ΔILT	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Index (β)	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$t_{o,\beta}$	335	341	347	353	359	365	371	377	383	389	395	401	407	413	419	425	431	437	443	449	455	461	467	473	486
IAT	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
ILT	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	13
RI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
ΔILT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
Index (β)	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
$t_{o,\beta}$	494	502	510	518	526	534	542	550	558	566	574	582	590	598	606	614	622	630	638	646	654	662	670	678	686
IAT	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
ILT	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
RI	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39	41	43	45	47	49	51	53	55	57
ΔILT	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Index (β)	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
$t_{o,\beta}$	694	702	710	718	726	734	742	750	758	766	774	782	790	798	806	814	822	830	838	846	854	862	870	878	886
IAT	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
ILT	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
RI	59	61	63	65	67	69	71	73	75	77	79	81	83	85	87	89	91	93	95	97	99	101	103	105	107
ΔILT	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

Appendix C-3 - Data of slowdown with constant inter-arrival time (Slight Transfer slowdown)

Index (β)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$t_{o,\beta}$	185	191	197	203	209	215	221	227	233	239	245	251	257	263	269	275	281	287	293	299	305	311	317	323	329
IAT	-	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
ILT	-	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
RI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ΔILT	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Index (β)	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$t_{o,\beta}$	335	341	347	353	359	365	371	377	383	389	395	401	407	413	419	425	431	437	443	449	455	461	467	473	479
IAT	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
ILT	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
RI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ΔILT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Index (β)	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
$t_{o,\beta}$	490	502	514	526	538	550	562	568	574	580	586	592	598	604	610	616	622	628	634	640	646	652	658	664	670
IAT	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
ILT	11	12	12	12	12	12	12	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
RI	5	11	17	23	29	35	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41
ΔILT	5	6	6	6	6	6	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Index (β)	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
$t_{o,\beta}$	676	682	688	694	700	706	712	718	724	730	736	742	748	754	760	766	772	778	784	790	796	802	808	814	820
IAT	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
ILT	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
RI	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41
ΔILT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Appendix C-4 - Data of slowdown with constant inter-arrival time (Serious Transfer slowdown)

Index (β)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$t_{o,\beta}$	185	191	197	203	209	215	221	227	233	239	245	251	257	263	269	275	281	287	293	299	305	311	317	323	329
IAT	-	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
ILT	-	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
RI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ΔILT	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Index (β)	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$t_{o,\beta}$	335	341	347	353	359	365	371	377	383	389	395	401	407	413	419	425	431	437	443	449	455	461	467	473	479
IAT	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
ILT	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
RI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ΔILT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Index (β)	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
$t_{o,\beta}$	500	524	548	572	596	620	644	652	660	668	676	684	692	700	708	716	724	732	740	748	756	764	772	780	788
IAT	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
ILT	21	24	24	24	24	24	24	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
RI	15	33	51	69	87	105	123	125	127	129	131	133	135	137	139	141	143	145	147	149	151	153	155	157	159
ΔILT	15	18	18	18	18	18	18	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Index (β)	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
$t_{o,\beta}$	796	804	812	820	828	836	844	852	860	868	876	884	892	900	908	916	924	932	940	948	956	964	972	980	988
IAT	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
ILT	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
RI	161	163	165	167	169	171	173	175	177	179	181	183	185	187	189	191	193	195	197	199	201	203	205	207	209
ΔILT	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

Appendix D-1 - Data of slowdown with random inter-arrival time (Slight Machine Slowdown)

Index (β)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$t_{o,\beta}$	185	190	195	200	205	210	215	220	225	230	239	244	250	269	274	279	284	289	294	299	305	317	330	366	377
IAT	-	2	3	2	2	2	2	13	5	12	11	2	9	19	2	3	5	2	2	2	20	12	13	36	11
ILT	-	5	5	5	5	5	5	5	5	5	9	5	6	19	5	5	5	5	5	5	6	12	13	36	11
RI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ΔILT	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Index (β)	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$t_{o,\beta}$	385	393	398	403	408	413	418	423	428	433	438	443	449	454	472	482	489	494	499	504	509	517	530	535	544
IAT	8	8	2	4	2	2	7	3	8	6	2	11	9	2	21	10	7	4	3	2	2	17	13	2	5
ILT	8	8	5	5	5	5	5	5	5	5	5	5	6	5	18	10	7	5	5	5	5	8	13	5	9
RI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
ΔILT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
Index (β)	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
$t_{o,\beta}$	555	560	565	570	586	591	596	601	606	611	616	628	645	651	667	672	677	682	687	692	697	702	707	712	717
IAT	14	4	2	8	17	5	5	4	2	6	6	14	17	6	16	2	2	2	2	5	5	3	2	2	2
ILT	11	5	5	5	16	5	5	5	5	5	5	12	17	6	16	5	5	5	5	5	5	5	5	5	5
RI	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
ΔILT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Index (β)	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
$t_{o,\beta}$	722	727	732	737	742	747	752	757	762	767	772	777	800	805	818	823	828	838	843	848	853	858	893	903	909
IAT	2	2	2	2	13	5	5	19	2	6	2	19	27	3	15	2	4	14	2	3	8	2	40	10	6
ILT	5	5	5	5	5	5	5	5	5	5	5	5	23	5	13	5	5	10	5	5	5	5	35	10	6
RI	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
ΔILT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Appendix D-2 - Data of slowdown with random inter-arrival time (Serious Machine Slowdown)

Index (β)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$t_{o,\beta}$	185	190	195	200	205	210	215	220	225	230	239	244	250	269	274	279	284	289	294	299	305	317	330	366	377
IAT	-	2	3	2	2	2	2	13	5	12	11	2	9	19	2	3	5	2	2	2	20	12	13	36	11
ILT	-	5	5	5	5	5	5	5	5	5	9	5	6	19	5	5	5	5	5	5	6	12	13	36	11
RI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ΔILT	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Index (β)	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$t_{o,\beta}$	385	393	398	403	408	413	418	423	428	433	438	443	449	454	472	482	489	494	499	504	509	517	530	535	547
IAT	8	8	2	4	2	2	7	3	8	6	2	11	9	2	21	10	7	4	3	2	2	17	13	2	5
ILT	8	8	5	5	5	5	5	5	5	5	5	5	6	5	18	10	7	5	5	5	5	8	13	5	12
RI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
ΔILT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
Index (β)	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
$t_{o,\beta}$	558	566	574	582	590	598	606	614	622	630	638	646	654	662	670	678	686	694	702	710	718	726	734	742	750
IAT	14	4	2	8	17	5	5	4	2	6	6	14	17	6	16	2	2	2	2	5	5	3	2	2	2
ILT	11	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
RI	7	10	13	16	8	11	14	17	20	23	26	22	13	15	7	10	13	16	19	22	25	28	31	34	37
ΔILT	0	3	3	3	-8	3	3	3	3	3	3	-4	-9	2	-8	3	3	3	3	3	3	3	3	3	3
Index (β)	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
$t_{o,\beta}$	758	766	774	782	790	798	806	814	822	830	838	846	854	862	870	878	886	894	902	910	918	926	934	942	950
IAT	2	2	2	2	13	5	5	19	2	6	2	19	27	3	15	2	4	14	2	3	8	2	40	10	6
ILT	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
RI	40	43	46	49	52	55	58	61	64	67	70	73	58	61	56	59	62	60	63	66	69	72	45	43	45
ΔILT	3	3	3	3	3	3	3	3	3	3	3	3	-15	3	-5	3	3	-2	3	3	3	3	-27	-2	2

Appendix D-3 - Data of slowdown with random inter-arrival time (Slight Transfer slowdown)

Index (β)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$t_{o,\beta}$	185	190	195	200	205	210	215	220	225	230	239	244	250	269	274	279	284	289	294	299	305	317	330	366	377
IAT	-	2	3	2	2	2	2	13	5	12	11	2	9	19	2	3	5	2	2	2	20	12	13	36	11
ILT	-	5	5	5	5	5	5	5	5	5	9	5	6	19	5	5	5	5	5	5	6	12	13	36	11
RI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ΔILT	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Index (β)	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$t_{o,\beta}$	385	393	398	403	408	413	418	423	428	433	438	443	449	454	472	482	489	494	499	504	509	517	530	535	540
IAT	8	8	2	4	2	2	7	3	8	6	2	11	9	2	21	10	7	4	3	2	2	17	13	2	5
ILT	8	8	5	5	5	5	5	5	5	5	5	5	6	5	18	10	7	5	5	5	5	8	13	5	5
RI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ΔILT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Index (β)	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
$t_{o,\beta}$	555	561	568	575	596	601	607	611	616	621	627	639	655	661	677	683	687	692	697	703	707	712	717	723	727
IAT	14	4	2	8	17	5	5	4	2	6	6	14	17	6	16	2	2	2	2	5	5	3	2	2	2
ILT	15	6	7	7	21	5	6	4	5	5	6	12	16	6	16	6	4	5	5	6	4	5	5	6	4
RI	4	5	7	9	14	14	15	14	14	14	15	15	14	14	14	15	14	14	14	15	14	14	14	15	14
ΔILT	4	1	2	2	5	0	1	-1	0	0	1	0	-1	0	0	1	-1	0	0	1	-1	0	0	1	-1
Index (β)	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
$t_{o,\beta}$	732	737	743	747	752	757	763	767	772	777	783	787	811	815	828	833	839	848	853	859	863	868	902	913	918
IAT	2	2	2	2	13	5	5	19	2	6	2	19	27	3	15	2	4	14	2	3	8	2	40	10	6
ILT	5	5	6	4	5	5	6	4	5	5	6	4	24	4	13	5	6	9	5	6	4	5	34	11	5
RI	14	14	15	14	14	14	15	14	14	14	15	14	15	14	14	14	15	14	14	15	14	14	13	14	13
ΔILT	0	0	1	-1	0	0	1	-1	0	0	1	-1	1	-1	0	0	1	-1	0	1	-1	0	-1	1	-1

Appendix D-4 - Data of slowdown with random inter-arrival time (Serious Transfer slowdown)

Index (β)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$t_{o,\beta}$	185	190	195	200	205	210	215	220	225	230	239	244	250	269	274	279	284	289	294	299	305	317	330	366	377
IAT	-	2	3	2	2	2	2	13	5	12	11	2	9	19	2	3	5	2	2	2	20	12	13	36	11
ILT	-	5	5	5	5	5	5	5	5	5	9	5	6	19	5	5	5	5	5	5	6	12	13	36	11
RI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ΔILT	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Index (β)	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$t_{o,\beta}$	385	393	398	403	408	413	418	423	428	433	438	443	449	454	472	482	489	494	499	504	509	517	530	535	540
IAT	8	8	2	4	2	2	7	3	8	6	2	11	9	2	21	10	7	4	3	2	2	17	13	2	5
ILT	8	8	5	5	5	5	5	5	5	5	5	5	6	5	18	10	7	5	5	5	5	8	13	5	5
RI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ΔILT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Index (β)	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
$t_{o,\beta}$	581	601	621	641	705	713	721	729	737	745	753	761	769	777	785	793	801	809	817	825	833	841	849	857	865
IAT	14	4	2	8	17	5	5	4	2	6	6	14	17	6	16	2	2	2	2	5	5	3	2	2	2
ILT	41	20	20	20	64	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
RI	30	45	60	75	123	126	129	132	135	138	141	137	128	130	122	125	128	131	134	137	140	143	146	149	152
ΔILT	30	15	15	15	48	3	3	3	3	3	3	-4	-9	2	-8	3	3	3	3	3	3	3	3	3	3
Index (β)	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
$t_{o,\beta}$	873	881	889	897	905	913	921	929	937	945	953	961	969	977	985	993	1001	1009	1017	1025	1033	1041	1049	1057	1065
IAT	2	2	2	2	13	5	5	19	2	6	2	19	27	3	15	2	4	14	2	3	8	2	40	10	6
ILT	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
RI	155	158	161	164	167	170	173	176	179	182	185	188	173	176	171	174	177	175	178	181	184	187	160	158	160
ΔILT	3	3	3	3	3	3	3	3	3	3	3	3	-15	3	-5	3	3	-2	3	3	3	3	-27	-2	2

Appendix E - Simulation result of response time Vs blocking location for ROIs

ROI1 ($T_s = 60$)		ROI2 ($T_s = 56$)	
Blocking locations	Response time	Blocking locations	Response time
4	12	8	24
8	16	14	32
20	28	26	30
24	28	30	26
37	23	40	16
41	19	44	12
55	13	54	10
ROI3 ($T_s = 56$)		ROI4 ($T_s = 60$)	
Blocking locations	Response time	Blocking locations	Response time
4	12	4	12
7	17	8	16
17	31	20	32
21	33	24	32
32	24	37	23
36	20	41	19
45	11	51	17
49	9	55	13

Appendix F – Implementation source code

File Name: Model.h

```
#ifndef MODEL_H

#define MODEL_H

#include "Element.h"

/** initialize process and prepare memory

@param:

time_standard        (Ts) Ideal time to travel from Inlet to Outlet

time_gap             (Tg) Min gap between two components

cap_element          No. of elements in this run

cap_ws               No. of work-stations in this run

@return

1:       OK

-1:      Failed

*/

extern "C" __declspec(dllexport) __stdcall int model_init(long int time_standard, long int time_gap, long int cap_element, long int cap_ws);


/** free the memory

@return

0:                       OK

*/

extern "C" __declspec(dllexport) __stdcall int model_end();
```

```
/** setup event chain

[please insert all events before "model_run()" function, or you need "model_reset()" to redo all the elements]

[the 0th event is the Inlet & the nth event is the Outlet]

@param

process_time:          processing time of the work-station (Tp,k)

transfer_time:         transfer time from the previous work-station to current work-station.

@return

x:      event_id (range: 0 to max_event_num)

-1:     Failed (out of bound) or (events have been set)

*/

extern "C" __declspec(dllexport) __stdcall int New_ws(long int process_time, long int transfer_time);


/*!

*@fn Create new component for the program, index of the component is generated

*@param[in] _it_i The clock time when the component arrives at the inlet

*@return 0 for success or -1 for fail

*/

extern "C" __declspec(dllexport) __stdcall long int New_Comp(long int _it_i);


/*!

*@fn Retrive the predicted component outlet time

*@param[in] _iComp_Index The index of the component
```

```
*@return x for success or -1 for fail        x means the predicted outlet time

*/

extern "C" __declspec(dllexport) __stdcall long int Get_Comp_Outlet_Time(long int
_iComp_Index);



/*!

*@fn Calculate the inter-arrival gap between the component and its previous one

*@param[in] _iComp_Index The index of the component

*@ return 0 for success or -1 for fail

*/

extern "C" __declspec(dllexport) __stdcall long int Set_Comp_Arrival_Gap (long int
_iComp_Index);



/*!

*@fn Retrive the inter-arrival gap of the component

*@param[in] _iComp_Index The index of the component

*@return x  for success or -1 for fail        x means the inter-arrival gap

*/

extern "C" __declspec(dllexport) __stdcall long int Get_Comp_Arrival_Gap(long int
_iComp_Index);



/*!

*@fn Retrive the total time delay of the component within the ROI

*@param[in] _iComp_Index The index of the component

*@ return x for success or -1 for fail        x means the time delay

*/
```

```
extern "C" __declspec(dllexport) __stdcall long int Get_Comp_Delay (long int
_iComp_Index);


/*!

*@fn identify location of the BBP

*@param[in] _iT_e The ideal inter-component arrival time

*@param[out] _iK_bbp The index of work-station just before the BBP

*@param[out] _iT_bbp The transfer time from its previous work-station to the BBP

*@return 0 for success or -1 for fail

*/

extern "C" __declspec(dllexport) __stdcall long int BBP_Identification (long int _iT_e,
long int& _iK_bbp, long int& _iT_bbp);


/*!

*@fn Segment an ROI from the production line

*@param[in] _iT_e The ideal inter-component arrival time

*@param[in] _iT_rc The tolerable maximum response time of blocking case

*@param[out] _iTs_Target The length of the segmented ROI

*@param[out] _ik_ws The largest index of the work-stations in the target ROI

*@param[out] _iT_position The length from the (_ik_ws)th work-station to the segment
point

*@return 0 for success or -1 for fail

*/

extern "C" __declspec(dllexport) __stdcall long int ROI_Segment(long int _iT_e, long int
_iT_rc, long int& _iTs_Target, long int& _ik_ws, long int& _iT_position);
```

```
/*!
*@fn Segment an ROI from the production line
*@param[in] _iT_e The ideal inter-component arrival time
*@param[in] _iT_rc The tolerable maximum response time of blocking case
*@param[out] _iTs_Target The length of the segmented ROI
*@param[out] _ik_ws The largest index of the work-stations in the target ROI
*@return 0 for success or -1 for fail
*/
extern "C" __declspec(dllexport) __stdcall long int ROI_Segment_WB(long int _iT_e, long int _iT_rc, long int& _iTs_Target, long int& _ik_ws);


/*!
*@fn Segment an ROI from the production line
*@param[in] _iTs_Target The length of the segmented ROI
*@param[in] _ik_ws The largest index of the work-stations in the target ROI
*@param[in] _iT_position The length from the (_ik_ws)th work-station to the segment point
*@return 0 for success or -1 for fail
*/
extern "C" __declspec(dllexport) __stdcall long int ROI_Reset(long int _iTs_Target, long int _ik_ws, long int _iT_position);


/*!
*@fn Get the last segmented ROI (to the outlet)
*@param[out] _iTs The length of the final ROI
*@return 0 for length>0 or -1 for length ==0
```

```
*/
extern "C" __declspec(dllexport) __stdcall long int Get_Final_ROI(long int& _iTs);


// local variables to use
static long int __ELEMENT_CAP = 0;
static long int __WS_CAP = 0;
static long int __ELEMENT_NUM = 0;
static long int __WS_NUM = 0;
static long int __TIME_S = 0;
static long int __TIME_G = 0;
static long int __TIME_BOTTLENECK = 0;
static WORKSTATION** workstations = NULL;
static Element** elements = NULL;


//varialbe used for ROI segmentation
static long int k_current = 0;


struct ROI
{
	long int WS_Qty;		//Quantity of Work-stations
	long int TG;		//Minimum gap between components
	long int* WS_PT;		//Pointer to the work-stations processing times array
	long int* TT;		//Pointer to the transfer times array
};
#endif
```

File Name: Element.h

```
#ifndef ELEMENT_H
#define ELEMENT_H
 class Element
{
public:
        int id;                         // Index of the component
        Element* prevElement;           // previous element
        long int time_start;            // start time
        long int time_standard;         // standard time;
        long int time_min_gap;          // Minimum gap time between components
        long int time_bottleneck;       // Processing time of bottleneck work-station
        long int time_gap;              // Time gap between previous components
        long int time_delay;            // Total delay caused on the component
        Element();                      // constructor
        virtual ~Element();             // destructor

        /*****************************************************************/
        /* Calculate the total delay on the component                    */
        /*****************************************************************/
        void calculate_delay();

        /*****************************************************************/
        /* Get the predicted outlet time of component                    */
```

```
        /*****************************************************************/
        long int get_time_out();

        /*!
        *@fn Retrive the total time delay of the component within the ROI
        *@param[in] _iComp_Index The index of the component
        *@ return x for success or -1 for fail        x means the time delay
        */
        long int Get_Comp_Delay ();

        /*!
        *@fn Calculate the inter-arrival gap between the component and its previous one
        *@param[in] _iComp_Index The index of the component
        *@ return 0 for success or -1 for fail
        */
        long int Set_Comp_Arrival_Gap ();

        /*!
        *@fn Retrive the inter-arrival gap of the component
        *@param[in] _iComp_Index The index of the component
        *@return x  for success or -1 for fail        x means the inter-arrival gap
        */
        long int Get_Comp_Arrival_Gap();
};
struct WORKSTATION
```

```
{
        long int id;
        long int process_time;          //processing time of the work-station
        long int transfer_time;         //transfer time from the previous ws to current one.
};
#endif
```

File Name: Model.cpp

```
#include <malloc.h>
#include <cstring>
#include <stdio.h>
#include <fstream>
#include "stdafx.h"
#include "model.h"
#include "Element.h"


extern "C" __declspec(dllexport) __stdcall int model_init(long int time_standard, long int time_gap, long int cap_element, long int cap_ws)
{
	if (__ELEMENT_NUM >0 && __WS_NUM>0)
	{
		model_end();
	}
	__TIME_S = 0;
	__TIME_G = time_gap;
	__ELEMENT_CAP = cap_element;
	__WS_CAP = cap_ws;
	__ELEMENT_NUM = 0;		// started from 1
	__WS_NUM = 0;			// started from 1
	elements = (Element**)malloc(sizeof(Element*) * __ELEMENT_CAP);
	workstations = (WORKSTATION**)malloc(sizeof(WORKSTATION*) * __WS_CAP);
	// set 0-th event as "point in"
	New_ws(0, 0);
	return 0;
}


extern "C" __declspec(dllexport) __stdcall int model_end()
{
	for (long int i=__ELEMENT_NUM;i>0;i--)
```

```
	{
		delete elements[i];
	}
	for (i=__WS_NUM-1;i>=0;i--)
	{
		delete workstations[i];
	}
	__ELEMENT_NUM = 0;
	__WS_NUM = 0;
	free(elements);
	free(workstations);
	return 0;
}

extern "C" __declspec(dllexport) __stdcall int New_ws(long int process_time, long int
transfer_time)
{
	long int index = __WS_NUM++;
	WORKSTATION* ws = new WORKSTATION;
	ws->id = index;
	ws->process_time = process_time;
	ws->transfer_time = transfer_time;
	workstations[index] = ws;

	__TIME_S += process_time + transfer_time;

	//update the value of bottleneck
	for(long int i=0; i<index; i++)
	{
		if(__TIME_BOTTLENECK < workstations[i]->process_time)
		{
			__TIME_BOTTLENECK = workstations[i]->process_time;
		}
```

```
	}

	//printf("after ws %ld, the bottleneck is %ld\r\n", __WS_NUM,
__TIME_BOTTLENECK);
	return index;
}

extern "C" __declspec(dllexport) __stdcall long int New_Comp(long int _it_i)
{
	long int index = ++__ELEMENT_NUM;
	Element* e = new Element();
	e->id = index;
	e->time_start = _it_i;
	e->time_standard = __TIME_S;
	e->time_min_gap = __TIME_G;
	e->time_bottleneck = __TIME_BOTTLENECK;

	if(e->id == 1)
	{
		e->prevElement = 0;
	}
	else
	{
		e->prevElement = elements[index - 1];
	}

	elements[index] = e;   // add this element to global element array

	return index;
}

extern "C" __declspec(dllexport) __stdcall long int Get_Comp_Outlet_Time(long int
_iComp_Index)
```

```
{
	return elements[_iComp_Index]->get_time_out();
}


extern "C" __declspec(dllexport) __stdcall long int Set_Comp_Arrival_Gap (long int
_iComp_Index)
{
	return elements[_iComp_Index]->Set_Comp_Arrival_Gap();
}


extern "C" __declspec(dllexport) __stdcall long int Get_Comp_Arrival_Gap(long int
_iComp_Index)
{
	return elements[_iComp_Index]->Get_Comp_Arrival_Gap();
}


extern "C" __declspec(dllexport) __stdcall long int Get_Comp_Delay (long int
_iComp_Index)
{
	return elements[_iComp_Index]->Get_Comp_Delay();
}


extern "C" __declspec(dllexport) __stdcall long int BBP_Identification (long int _iT_e,
long int& _iK_bbp, long int& _iT_bbp)
{
	long int ret = 0;
	long int Count;
	long int k_bbp=0, T_k_bbp =0;
	long int Sum_Tk=0;


	if(!_iT_e)
	{
		return -1;
	}
```

```
	Count = (__TIME_S - _iT_e) / _iT_e;

	for(k_bbp = 0; k_bbp <= __WS_NUM; k_bbp++)
	{
		Sum_Tk = Sum_Tk + workstations[k_bbp]->transfer_time;
		for(T_k_bbp=0; T_k_bbp<=workstations[k_bbp+1]->transfer_time;
T_k_bbp++)
		{
			if(((T_k_bbp + Sum_Tk + __TIME_G)/__TIME_G + k_bbp) ==
Count)
			{
				_iK_bbp = k_bbp;
				_iT_bbp = T_k_bbp;
				return 0;
			}
		}
	}

	return -1;
}

extern "C" __declspec(dllexport) __stdcall long int ROI_Segment(long int _iT_e, long int
_iT_rc, long int& _iTs_Target, long int& _ik_ws, long int& _iT_position)
{
	long int k_bbp, T_start, T_end, intersept;
	long int T_s;
	long int T_rb;
	long int ret = -1;
	long int ws_index = 0;
	for(k_bbp=0; k_bbp <__WS_NUM-1; k_bbp++)
	{
		long int j;
```

```
                //Calculate the start point and end point of piecewise function, like "A",
"B", "C" point
                if(k_bbp == 0)
                {
                        T_start = 0;
                }
                else
                {
                        T_start = (k_bbp-1)*_iT_e;
                }
                intersept = k_bbp*__TIME_G;
                for(j=0; j<=k_bbp; j++)
                {
                        T_start += (_iT_e/__TIME_G)*workstations[j]->transfer_time;
                        intersept -= workstations[j]->process_time;
                }
                if(k_bbp == 0)
                {
                        T_end = T_start + (_iT_e/__TIME_G)*workstations[k_bbp+1]-
>transfer_time;
                }
                else
                {
                        T_end = T_start + _iT_e +
(_iT_e/__TIME_G)*workstations[k_bbp+1]->transfer_time;
                }

                for(T_s=T_start; T_s<=T_end; T_s++)
                {
                        if(T_s <= __TIME_S)
                        {
                                T_rb = (1-(double)__TIME_G/(double)_iT_e)*T_s +
intersept;
```

```
                        long int T_workstation =0, Temp_T_workstation =0;
                        for(long int k=0; k<=ws_index; k++)
                        {
                                T_workstation += workstations[k]->process_time +
workstations[k]->transfer_time;
                        }
                        Temp_T_workstation = T_workstation;
                        T_workstation += workstations[ws_index+1]-
>transfer_time;

                        if(T_workstation == T_s) //arrive at an workstation
                        {
                                T_s = T_s + workstations[ws_index+1]-
>process_time;
                                ws_index++;

                                if(_iT_rc > T_rb && _iT_rc <((1-
(double)__TIME_G/(double)_iT_e)*T_s + intersept))
                                {
                                        _iTs_Target = T_s;
                                        _ik_ws = workstations[ws_index]->id;
                                        _iT_position = 0;

                                        k_current = ws_index;
                                        ret = 0;
                                }
                        }
                        else if(T_rb == _iT_rc)
                        {
                                _iTs_Target = T_s;
                                _ik_ws = workstations[ws_index]->id;
                                _iT_position = T_s - Temp_T_workstation;

                                k_current = ws_index;
```

```
                                        ret = 0;
                                    }
                                }
                            }
                        }
                        return ret;
}

extern "C" __declspec(dllexport) __stdcall long int ROI_Segment_WB(long int _iT_e, long int _iT_rc, long int& _iTs_Target, long int& _ik_ws)
{
        long int k_bbp, T_start, T_end, intersept;
        long int T_s;
        long int T_rb;
        long int ret = -1;
        long int ws_index = 0;
        for(k_bbp=0; k_bbp <__WS_NUM-1; k_bbp++)
        {
                long int j;

                //Calculate the start point and end point of piecewise function, like "A", "B", "C" point
                if(k_bbp == 0)
                {
                        T_start = 0;
                }
                else
                {
                        T_start = (k_bbp-1)*_iT_e;
                }
                intersept = k_bbp*__TIME_G;
                for(j=0; j<=k_bbp; j++)
                {
```

```
                        T_start += (_iT_e/__TIME_G)*workstations[j]->transfer_time;
                        intersept -= workstations[j]->process_time;
                }
                if(k_bbp == 0)
                {
                        T_end = T_start + (_iT_e/__TIME_G)*workstations[k_bbp+1]-
>transfer_time;
                }
                else
                {
                        T_end = T_start + _iT_e +
(_iT_e/__TIME_G)*workstations[k_bbp+1]->transfer_time;
                }

                for(T_s=T_start; T_s<=T_end; T_s++)
                {
                        if(T_s <= __TIME_S)
                        {
                                T_rb = (1-(double)__TIME_G/(double)_iT_e)*T_s +
intersept;

                                long int T_workstation =0, Temp_T_workstation =0;
                                for(long int k=0; k<=ws_index; k++)
                                {
                                        T_workstation += workstations[k]->process_time +
workstations[k]->transfer_time;
                                }
                                Temp_T_workstation = T_workstation;
                                T_workstation += workstations[ws_index+1]-
>transfer_time;

                                if(T_workstation == T_s) //arrive at an workstation
                                {
```

```
                                        T_s = T_s + workstations[ws_index+1]-
>process_time;
                                        ws_index++;

                                        if(_iT_rc > T_rb && _iT_rc <((1-
(double)__TIME_G/(double)_iT_e)*T_s + intersept))
                                        {
                                                _iTs_Target = T_s;
                                                _ik_ws = workstations[ws_index]->id;

                                                k_current = ws_index;
                                                ret = 0;
                                        }
                                }
                                else if(T_rb == _iT_rc)
                                {
                                        _iTs_Target = Temp_T_workstation;
                                        _ik_ws = workstations[ws_index]->id;

                                        k_current = ws_index;
                                        ret = 0;
                                }
                        }
                }
        }
        return ret;
}

extern "C" __declspec(dllexport) __stdcall long int ROI_Reset(long int _iTs_Target, long
int _ik_ws, long int _iT_position)
{
        long int i;
```

```
	if(_iTs_Target >= __TIME_S || k_current >= __WS_NUM)
		return -1;

	for(i=1; i<__WS_NUM-k_current; i++)
	{
		workstations[i]->id = workstations[i+k_current]->id;
		workstations[i]->process_time = workstations[i+k_current]-
>process_time;
		workstations[i]->transfer_time = workstations[i+k_current]-
>transfer_time;
	}
	workstations[1]->transfer_time = workstations[1]->transfer_time - _iT_position;

	for(i=__WS_NUM-k_current; i<__WS_NUM; i++)
	{
		workstations[i]->id = 0;
		workstations[i]->process_time = 0;
		workstations[i]->transfer_time =0;
	}
	__TIME_S = __TIME_S - _iTs_Target;
	__WS_NUM = __WS_NUM - k_current;

	k_current =0;

	return 0;
}

extern "C" __declspec(dllexport) __stdcall long int Get_Final_ROI(long int& _iTs)
{
	_iTs = __TIME_S;

	if(_iTs ==0)
	{
```

```
                return -1;
        }
        else
        {
                return 0;
        }
}
```

File Name: Element.cpp

```
#include "stdafx.h"
#include <malloc.h>
#include <assert.h>
#include <stdlib.h>
#include "Element.h"

Element::Element()
{
	this->id = -1;
	this->prevElement = 0;
	this->time_start = -1;
	this->time_standard = -1;
	this->time_min_gap = -1;
	this->time_bottleneck = -1;
	this->time_gap = -1;
	this->time_delay = -1;
};

Element::~Element()
{
};

void Element::calculate_delay()
{
	//The first element, no time delay
	if(prevElement == 0)
	{
		time_delay = 0;
		return;
	}
```

```
	if(prevElement && prevElement->time_delay <0)
	{
		//If the previous one is not calculated, calculate it first
		prevElement->calculate_delay();
	}
	//calculate the gap of previous component
	long int ret = Set_Comp_Arrival_Gap();
	if(ret != 0)
	{
		return;
	}

	long int actual_time_delay = time_gap - prevElement->time_delay;
	if(actual_time_delay >= time_bottleneck)
	{
		time_delay = 0;
	}
	else if(actual_time_delay > time_gap)
	{
		time_delay = time_bottleneck - actual_time_delay;
	}
	else
	{
		time_delay = prevElement->get_time_out() + time_bottleneck -
(time_start + time_standard);
	}
};

long int Element::get_time_out()
{
	if(time_delay < 0)
	{
		//Did not calculate before
```

```
                calculate_delay();
        }
        return time_start + time_standard + time_delay;
};

long int Element::Get_Comp_Delay()
{
        if(time_delay < 0)
        {
                calculate_delay();
        }
        return time_delay;
};

long int Element::Set_Comp_Arrival_Gap ()
{
        long int ret = 0;
        if(prevElement)
        {
                time_gap = time_start - prevElement->time_start;
        }
        else
        {
                ret = -1;
        }
        return ret;
}

long int Element::Get_Comp_Arrival_Gap()
{
        return time_gap;
}
```

File Name: dllExport.def

```
; dllExport.def : Declares the module parameters for the DLL.
        LIBRARY      "ROI Prediction"
        EXPORTS
; Explicit exports can go here
        model_init
        model_end
        New_ws
        New_Comp
        Get_Comp_Outlet_Time
        Set_Comp_Arrival_Gap
        Get_Comp_Arrival_Gap
        Get_Comp_Delay
        BBP_Identification
        ROI_Segment
        ROI_Segment_WB
        ROI_Reset
        Get_Final_ROI
```