THE HONG KONG POLYTECHNIC UNIVERSITY

DEPARTMENT OF INDUSTRIAL AND SYSTEMS ENGINEERING

SYSTEM DYNAMICS IDENTIFICATION THROUGH ELEMENT FLOW REASONING

TANG CHI SHING

A thesis submitted in partial fulfillment of the requirements for Degree of Doctor of Philosophy

June 2009

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 test.

_____ (Signed)

TANG Chi Shing (Name of student)

Abstract of thesis entitled

‘System Dynamics Identification Through Element Flow Reasoning’

Submitted by Tang Chi Shing

For the degree of Doctor of Philosophy

At The Hong Kong Polytechnic University in 2009

ABSTRACT

Dynamic issues always occur in systems such as production lines, logistic systems, traffic systems, etc. To have a clear understanding of the overall operating status of a system is important if the appropriate response is to be triggered because even some small fluctuations can have a significant influence on the associated functions. However, it is very difficult to replicate real-life operations exactly by any modeling technique. In this research, a methodology has been established to reason the system’s behaviors in a dynamic manner by examining the characteristics of selected element flow streams. The concept is based on the top-down approach that contains three stages. The first stage is to identify the element flow streams such that they are able to represent the system and the second stage is about the transformation of measuring data into graphical patterns based on the proposed algorithms. In the final stage, a reasoning scheme is employed to extract the information embedded in the graphical patterns so that the system condition can be understood.

For experimental purpose, a software program has been coded with the use of DLL to provide better portability. Typical cases such as Overflow, Normal, Slowdown, Blocking, and Unknown Event Occurrence were tested. It was observed that the

proposed methodology was able to assist the system condition recognition. In terms of hardware requirements, only simple counting devices are needed and it is comparable to the Japanese's ANDON system but the human intervention factor can be reduced. To certain extent, the decision making process will have a good potential to be further automated with the application of the proposed methodology.

PUBLICATIONS ARISING FROM THE THESIS

Journals

1. Tang, C.S. and Chan, C.Y. (2006), 'Generic Mathematical Modelling for Monitoring Fabrication Flow Line', *Annual Journal of IIE (Hong Kong)*, V26, pp. 1-10.
2. Tang, C.S., Chan, C.Y. and Yung, K.L. (2007), 'Development of Dynamic Flow Line Monitoring Technique to Enhance System Transparency', *International Journal of Technology Intelligence and Planning*, V3, N(1), pp. 96-106.
3. Tang, C.S., Chan, C.Y. and Yung, K.L. (2007), 'Formation of a Generic Technique for Manufacturing Systems Monitoring', *International Journal of Technology Management*, V38, N(4), pp. 392-423.
4. Tang, C.S., Chan, C.Y. and Yung, K.L. (2007), 'The Development of a Generic Technique for Flow Line Monitoring', *IAENG - Engineering Letters*, V14, N(1), pp. 67-71.
5. Tang, C.S., Chan, C.Y., Yung, K.L., and Liu, H.R. (2007), 'The Formulation of System Monitoring Technique by Pattern Recognition', *Annual Journal of IIE (Hong Kong)*, V27, pp. 67-74.

Conferences

1. Tang, C.S., 'The Formulation of Generic System Dynamics Reasoning Approach', *Proceedings of 11th Annual International Conference on Industrial Engineering Theory, Applications and Practice*, (IJIE2006). Nagoya, Japan, October 2006, pp. 97-102.
2. Tang, C.S., Chan, C.Y., Yung, K.L., 'A Generic System Monitoring Technique by Using Similarity Recognition on the Flowing Entity Pattern', *Third International Conference on Natural Computation*, (ICNC 2007), Haikou, China, August 2007, pp.389-393.
3. Tang, C.S., Chan, C.Y., Yung, K.L., 'Application of pattern matching in dynamic system diagnostic', *Proceedings of the 8th Asia Pacific Industrial Engineering & Management System and 2007 Chinese Institute of Industrial Engineers Conference*, (APIEMS & CIIE Conference 2007), Kaohsiung, Taiwan, December, 2007, paper ID 196.

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere thanks to my supervisor, Dr. C.Y. Chan for his support, guidance and encouragement, which made this research project possible. Dr. C.Y. Chan has been a source of perceptive and critical comment, and moral support, through the project. His enthusiasm and discipline gave me strength and encouragement. I would like to thank my co-supervisor, Prof. K.L. Yung for his suggestions and comments.

I would also like to thank my parents for the support they provided me through my entire life and in particular, I must acknowledge my wife, without her love, encouragement and editing assistance, I would not have finished this thesis.

The work described in this research project was substantially supported by a grant from the Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University and the Research Grants Council of the Hong Kong Special Region.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Chapter One –Introduction to System Monitoring and Research Objective

1.1 Background of System Monitoring

The essential interactions between the control (or management) and the actuation (or operation) units are on the basis of the information exchange. Monitoring of the activities plays an important role in providing necessary information to the control unit. Generally speaking, there are two quite distinct styles of system monitoring, the Western and the Japanese styles. The former is the well-known Computer Integrated Manufacturing (CIM) approach while the later is based on the ANDON philosophy. However, one thing they have in common is that they both aim at obtaining good transparency in operating conditions. **Figure 1–1** shows a schematic outline of these two streams. In terms of CIM approaches, they rely on sensors to capture signals such as temperature, pressure, flow rate, frequency etc. from physical media. Then, these primary signals are refined into readable data by going through signal conditioning processes. Finally, data is transformed into meaningful information by employing dedicated mathematical models so that users can get a good picture of the system. On the other hand, the ANDON philosophy needs human involvement to judge the healthiness of a system. This is usually based on someone’s experience and is done by switching colored lights (usually, red, amber and green lights) on a pole, which is attached to a workstation. The main difference between CIM and ANDON is that ANDON provides a simple and flexible way to reflect the system statuses in a generic manner and it can be applied to monitor most systems. Unlike the CIM, where each station usually requires specific setups in association with a method to interpret the incoming data in order to provide the necessary information for actions. Consequently, the portability of CIM is quite low and new challenges will always be encountered whenever there is something new to be introduced to the system. Moreover, a later

modification can also be awkward since the interrelationships are so complex and happenings cannot be easily isolated. Clearly, the ANDON approach is much simpler and is able to get rid of problems such as complicated setups and the development of methods to digest the collected data. Moreover, it is a generic method that can be applied to monitor virtually every plant. However, it requires human intervention and thus, the results obtained can be inconsistent. Now, the question is whether it would be possible to extract and combine the advantages of both CIM and ANDON. The main objective of this research is to investigate the flowing characteristics of a specific element type in a system and the associated meanings behind these characteristics with the intention of extracting the required information to assist the understanding of the system's condition. With the knowledge of how to work out the material flow against time with respect to system behaviors, the inconsistency problem of the ANDON system can be resolved and the advantage of automatic reasoning from the CIM would be gained. Hence, the outcomes of this research will contribute to the monitoring of dynamics systems. It is expected that this concept can also be adopted in different fields such traffic flow, material movements and so on.

Apart from the CIM and the ANDON approaches, it is obvious that information regarding the fabrication situation is very important to a manufacturing organization because any fluctuation in the process can affect the associated logistics. Classically, the process condition can also be reflected indirectly by monitoring the quality level. A sound statistical approach needs a good documentation system to support it and today, there are commercial software packages available. The shortcoming of the Statistic Process Control (SPC) approach is the information gathered is usually piecemeal as only details on stations are collected and this may not explicitly

represent the overall effect on the system. Furthermore, the data processing time is usually lengthy because normally only daily quality control reports are provided. Perhaps, an online quality tracking system will be a positive move but the problem is neither concern with data collection method nor with product quality. The crucial point is to have a better overview of how the whole system is functioning and product quality can only tell this obliquely; good quality with a low output rate may indicate that there is a problem somewhere.

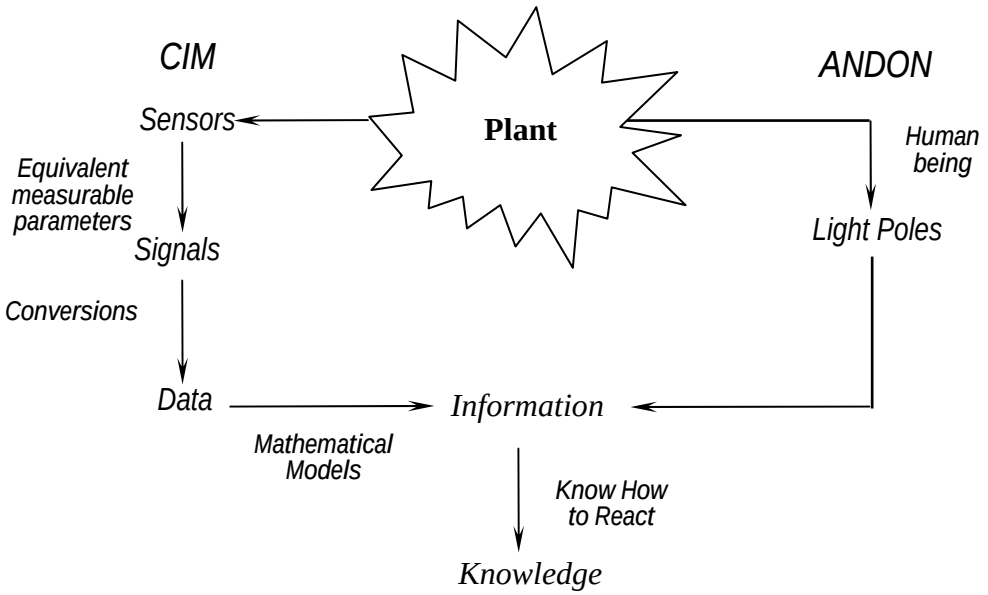


Figure 1-1. CIM and ANDON Information

1.2 Interpretation of System Dynamics

To understand a system, we start with a global sense of it. Regarding the dynamic behaviors of a system, it is usually composed of some moving entities. This can be categorized as some forms of “element flow”. For examples, there are flows such as the material flow in a manufacturing plant, the vehicles moving in a city, the blood circulating in our body, etc. Understanding how to interpret the variations of these flows leads to the fact that the conditions of the associated system can be made known.

Then, appropriate corrective action can be taken.

It is not difficult to observe that a system needs “energy” to keep it working. Then, the immediate question is how to get the required “energy” to maintain the system. In the macro view, the universal “energy” is benefits gained through activities performed by the system. However, the “energy” consumption has to be regulated in order to ensure the effectiveness of the system and this brings out the fact that the control unit needs the support of relevant “information” is timely so that appropriate decisions can be made. Simply stated, there are three foundation flows in a system: element, information and energy flows, and the overview of their interrelationships is delineated in **Figure 1–2**.

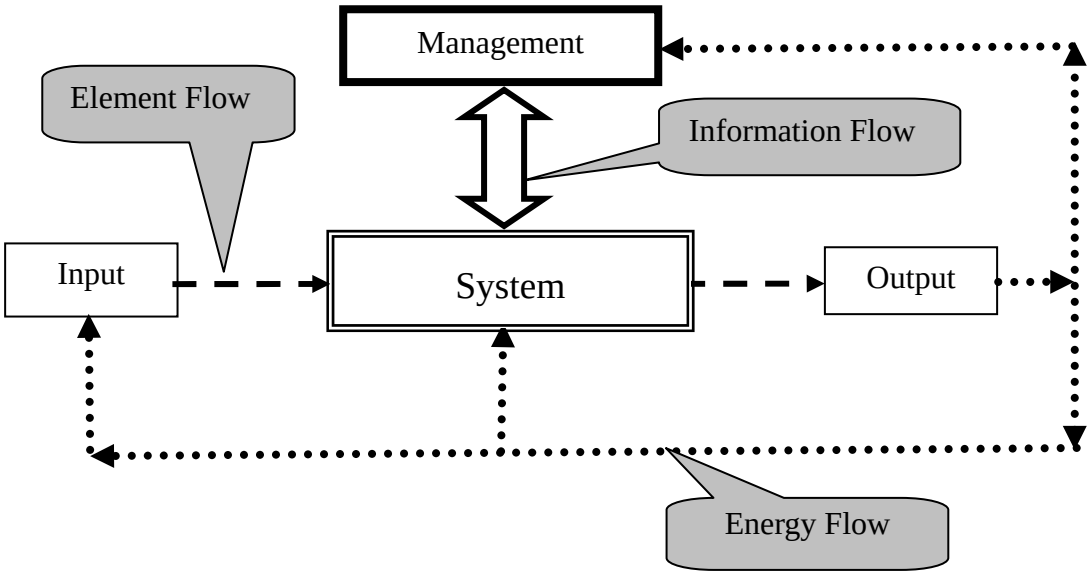


Figure 1-2. Three Basic Flow Types in a System

To further elaborate their relations, the basic function of a system is to add value to the flowing element from its input into the system all the way through a series of operations to the output. This can be viewed as the input of raw material into a

manufacturing system to produce products through a series of processes with the purpose of gaining the necessary energy (say, money) to maintain the system alive since the financial support is definitely crucial to keep it going. Put in another way, these two flows can also be regarded as gases (elements) burning in an engine to generate the thrust (energy) to keep an airplane flying. Nevertheless, these two flows will not work properly without suitable regulations and this brings out the importance of the third flow that is the information flow. As a matter of fact, it is certainly necessary to provide channels to enable the control unit to “see” the system so that the correct steering can be done, over time. In reality, effective information flow is very significant for running a system successfully.

1.3 System Reasoning by Signature Recognition

Fundamental approaches in system monitoring have been discussed, apart from the engineering control that is not the interest of this research. The elemental philosophy of this research is that we believe the condition of a system can be analogized by studying the element flow signatures in the system. In fact, there are examples that show how conditions can be made known by catching the coupled signatures. In the following sections, three examples will be presented to strengthen this standpoint: the Traditional Chinese Pulse Diagnosis and the In-Process Monitoring of Inertia Welding. Although we are not particularly fascinated by the actual implications of each example, they all illustrate how the signature recognition technique works in extracting information in a complex system.

Traditional Chinese Medical Science diagnoses illnesses by examining the patient’s

blood pulse on the wrists with Deep/Light touch on certain locations, **Figure 1–3** shows the recognized measuring positions on both wrists. They are named: Chy, Guan and Chun. A deep press on the left hand of these three positions can diagnose the health conditions of heart, liver and the pair of kidneys. In the same way, conditions of small intestine, gall bladder and bladder can be diagnosed by lightly pressing on the same hand. In contrast, similar positions on the right hand are to diagnose the lung, spleen and pericardium using a deep press. By the same token, large intestine, stomach and triple heater (‘San Jiao’ in Chinese) can be checked by a light touch on the right wrist. Traditionally, Chinese Doctors exercise their experiences in measuring the pulse strengths and pulse patterns from a patient to identify his sickness through their fingertips but this method depends greatly on the skill of a doctor and the diagnosis may not be always consistent. In view of this, Traditional Chinese Pulse Diagnosis (TCPD) method is developed to assist in the detection of the patients’ sickness by means of scientific approach. **Figure 1–4** shows a normal human blood pulse wave. The wave form could be distorted into certain shapes if an illness is present in the patient. By analyzing the pattern obtained, a person’s illness can be diagnosed. Some typical illnesses, Rhinitis, Asthma, Hepatitis and Laryngitis with their particular pulse wave patterns are given in **Table 1–1**.

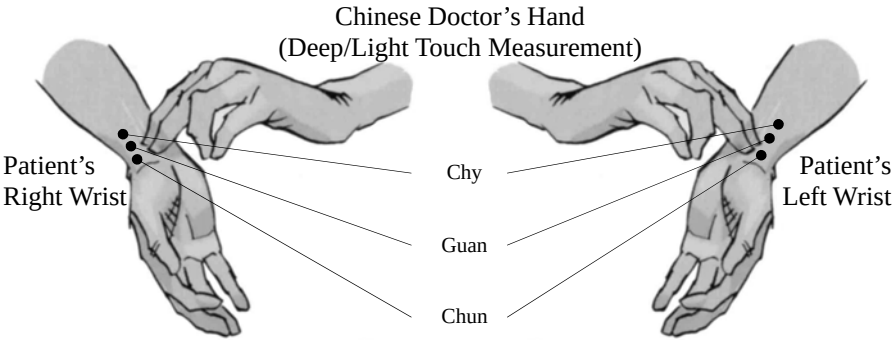


Figure 1-3. The Wrist Pulse Measuring Positions

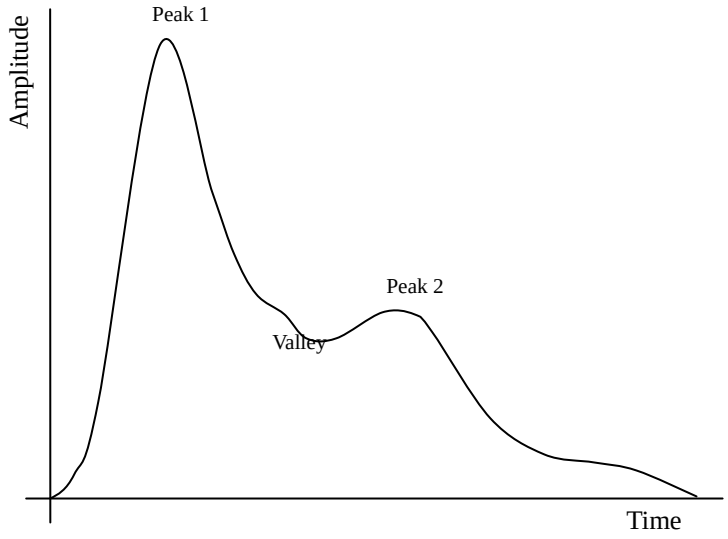
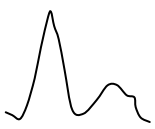
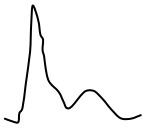
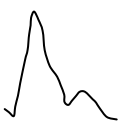


Figure 1-4. The Blood Pulse Signature

Table 1-1. Illness Blood Pulse Signatures

	Healthy	Rhinitis	Asthma	Hepatitis	Laryngitis
Pulse Signature					

The In-Process Monitoring for Inertia Welding example applied the signature recognition technique, with the help of a neural network, to identify the signature pattern from an inertia welding process of turning shafts. The research studied the acoustic signatures to verify the welding situations. **Figure 1–6** shows the normal acoustic signatures in the voltage measured. Usually, the wave form is relatively steady at the beginning stage since the sound is generated from the turning shaft on the left hand side only. The turning shaft then approaches the stationary shaft on the right, and the amplitude and the frequency increase significantly because the inter-surfaces of both shafts make contact and this is the moment of attack. Afterward, the amplitude decreases because the shaft material melts and decays. At the transitional stage, heated and deformed material is expelled from the interface between the two shafts since the turning energy is transformed and the acoustic wave decreases gradually until it becomes steady which means that the shafts have been welded together.

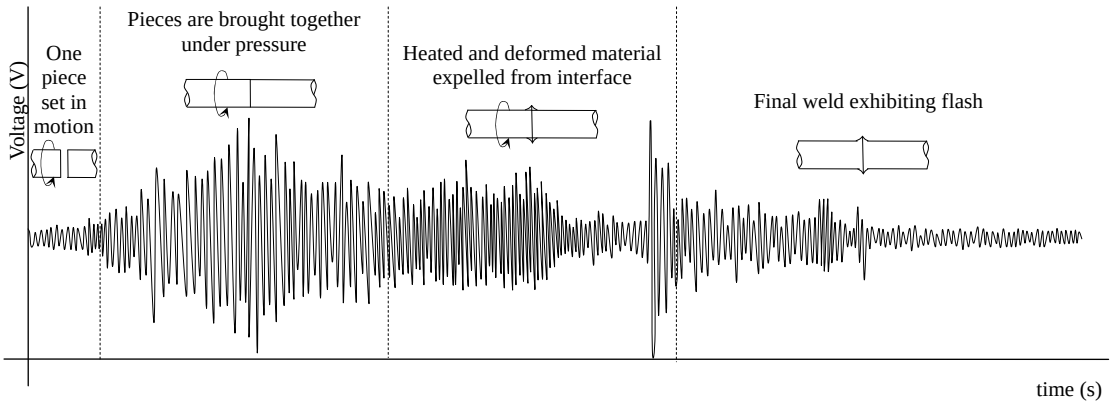


Figure 1-5. Inertia Welding Signature Recognition

This typical acoustic wave signature model was obtained by taking many experimental measurements so that if the wave signature emitted from a process does not match this pattern it can be said to be abnormal.

1.4 Research Objective

The main goal of this research is to align the analysis technique in system monitoring with the aim of establishing a generic approach with reference to the concept of ANDON. However, this requires the identification of what type of signals are to be investigated. Then, the special characteristics of the selected signal type will be investigated. The methodology used to relate the incoming signal to the system functioning condition is another challenge that needs to be addressed.

Indeed, benefit would be gained if a generic method to interpret the system behaviors can be developed and be incorporated into system monitoring. It is expected that such a methodology can work regardless of the properties of objects and therefore, applications are potentially able to extend to all types of measurable element moving systems. Then, by analyzing the variations of targeted moving elements, it can be decided if the system is performing as expected or not. To be more precise, the main objectives to be addressed in this research are:

- To establish a modular data collection framework that can be utilized generically for monitoring a dynamics system in a top-down[1] manner;
- To formulate algorithms in association with the proposed framework to transform the incoming data into useful information;
- To setup a method to analyse the obtained pattern so that the system operating status can be reflected.

[1] The ‘top-down’ approach for system monitoring here means looking at a system as a whole and identifies element flow branches to collect the required system information.

1.5 Research Contributions

It is anticipated that the proposed technique is not only capable of telling the condition of a system but will also be able to identify the nature of the problems and this will help to spot the sources of trouble more efficiently. At the end of this research, the software functions developed to verify the concept can be employed for other applications and therefore, it is anticipated that the functions will be compiled into a dynamic link library. The success of this research will have a significant impact on the system monitoring field as this approach is not tailor made for a particular system. In fact, it is aimed to be capable of providing monitoring of all types of systems where the condition of the system can be represented by selected detectable elements flowing in the system. It can be applied to systems such as transportation systems, manufacturing systems, customer service systems, etc. as they all exhibit this sort of flow. Moreover, the idea of examining a dynamics system as a whole by means of a modular approach can also benefit future developments in system monitoring.

2 Chapter Two – Developments of System Monitoring

2.1 Literature Review

The prime aim of this research is to formulate a system monitoring methodology in a top-down manner based on the foundation outlined in **Chapter 1**. The basic philosophy is comparable to the diagnostic method practiced by a traditional Chinese doctor who diagnoses sicknesses by examining the “blood pulse” and the perception of overall harmony which is essential to achieving real health. The latter also aligns with the Japanese Jidoka philosophy (Monden, 1998). It treats the system as a whole and relies on some information to detect the occurrence of any abnormality. In this chapter, the concept of system dynamics will be also explored with regard to a top -down manner. Inspection techniques will be reviewed because it is essential to identify a suitable tool to assist in the capturing of the system information which will later be analyzed.

2.2 Development of System Dynamics

In fact, a system can be interpreted as an integrated set of elements that accomplish a defined objective. This set of elements is usually in a changing status with regard to time (McKinney *et al.*, 2004). “Managing a system” is a process of getting activities completed efficiently and effectively through people. In reality, everyone applies certain soft skills to tackle their challenges either in personal affairs or in job related matters. We often rely on our own experience to resolve a problem and this can be referred to as empirical management (Hagoort 2003; Green, 2007). On the other hand, one may also solve a problem by applying a quantitative approach and this is regarded

as scientific management (Taylor, 2003). In the late 1950s, Forrester (Forrester, 1958) introduced the industrial dynamics philosophy to study the information feedback characteristics of an industrial or a business activity to show how policies, decisions and delays interact to influence the success of an enterprise. The Massachusetts Institute of Technology (MIT) termed it “Industrial Dynamics”. Later, it was renamed “Systems Dynamics” because the focus shifted to urban, economic, business, medical, ecological, and human systems in the late 60s (Forrester, 1961). Enterprise management is usually concerned with the dynamic interrelations of classified functional areas (e.g., marketing, investment, research, personnel, production, accounting, and etc.) and moving entities (e.g., money, orders, products, workers, facilities and so on). Formulating a relational model to observe the flow networks is the state of art of system dynamics (Zhu, 1996; Calvin, 2002). Although knowing every detail can be helpful, the management should devote themselves more to the strategic level decisions rather than to the routine work. Many scholars have proposed various approaches in dynamics systems monitoring including operational research (OR) approaches, computer integrated manufacturing (CIM), expert systems (ES), etc. (Stevenson, 2002; Benton and Shin, 1998; Tseng *et al.*, 1999; Ovacik and Uzsoy, 1993). One thing they have in common is that these approaches mainly focus on material flows together with operating facilities and the general objective is usually on getting improvement in efficiency. In practice, it is not easy to have good understanding of all the details in order to build a scientific model which works for an entire organization, as this would be far too complicated so the goal would be unreachable. To overcome this problem, small dedicated models are constructed such that each of them deals with one specific domain. But this may bring problems on later modifications once the external connections have been established for quite a

time. Furthermore, if there is any change to be made in the model, engineers are required to figure out clearly how this model functions and this may not be too straightforward if the existing approach is very detailed. Consequently, the Japanese scholars have thought about revising the concept scientific management to get rid of those complications of building a rational model. Nevertheless, they have mixed the scientific with the empirical knowledge from dedicated fieldworkers in various areas to form a more humanistic management philosophy and this is the Toyota Production System (TPS). ANDON is an indispensable building element of the TPS, it is a colored lights warning scheme that is triggered manually based on human experience. The aim is to reflect the workmanships of each of the attached facilities (Robert *et al.*, 2003). The advantage is clearly that this can greatly reduce the complexity of establishing system level monitoring.

2.3 System Data Collection

In order to be in keeping with the system level view, it is essential to gear oneself to the top-down approach way of thinking. This section looks into data collection because it is not difficult to see that homogenizing the data collection technique will largely reduce the complication of system maintenance.

In 1902, Mr. Sakichi Toyoda initiated the Jidoka philosophy that stops and resets the loom machines if any abnormality is detected (Mass and Robertson; 1996). Jidoka is one of the main pillars of the TPS while another contribution is the famous Just-In-Time (JIT) philosophy (Ono, 1988). ANDON is a component of the Jidoka that provides a platform for abnormality alerts (Li and Blumenfeld, 2005) and

Poka-yoke is another important component that encourages the implementation of simple and low-cost devices to detect abnormality and stops any process which produces defects (Dennis, 2002). A good Poka-yoke setup should satisfy the ideas of: simple with long life, low maintenance, high reliability and low cost under workplace conditions. In terms of monitoring a system, knowing whether the system is behaving abnormally or not is surely necessary. However, the definition of abnormalities may be difficult and consistency in detecting such abnormalities would rely on matching, which might be dubious.

2.4 The Poka-yoke Inspection Modes and Definition of Defect

The Statistical Process Control (SPC) principle states that a healthy process always entails a certain percentage of defects. For example some items out of the $\pm 3\sigma_s$ range can be tolerated and a process with the C_{pk} equals to 1.33/1.67 is considered to be good enough (Fujimoto, 1997). In 1910s, Mr. Sakichi Toyoda argued that defects from a process can be reduced to zero if the necessary corrective actions can be taken in time. Jidoka philosophy provides a perspective on errors and defects in which error is the source of an abnormality that may cause a defect so it is crucial to keep all operations out of error. Poka-yoke is the principle of helping people to “do things right the first time” and it encourages the use of simple low-cost devices to detect an abnormal situation before it occurs; or once it occurs, to stop the process in order to prevent any defect (Chen, 1994). Generally, Poka-yoke is employed in manufacturing but it has proved possible to apply in other areas such as logistics (e.g., order and invoice processing) and in hospitals (e.g., drug dispensing). **Figure 2–1**, shows the three Poka-yoke inspection modes: Judgment Inspection, Information Inspection and Source Inspection.

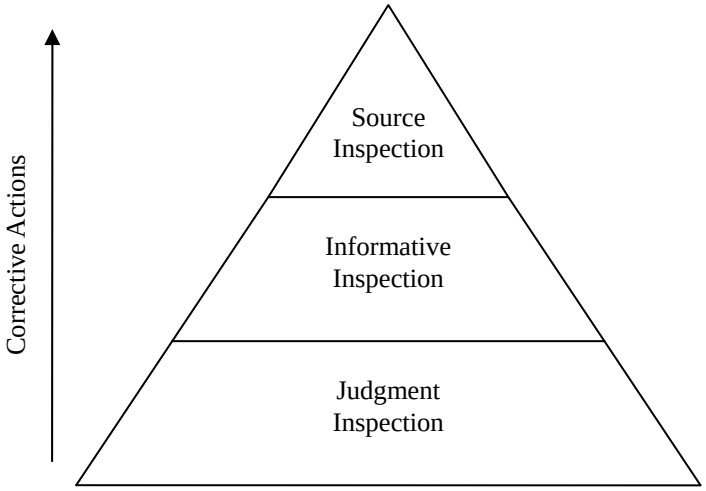


Figure 2-1. The 3 Poka-yoke Inspection Modes

The Judgment Inspection is the basic mode. It is a "**go or no-go**" inspection that stops the process to prevent the occurrence of defects due to some sort of error. This kind of inspection entails little root cause analysis or feedback to the source of errors. The judgments are always performed by workers in the context of their empirical experiences and it gives a fast response to the system. Informative Inspection lies at the middle level and is designed to discover defects and given feedback to the error source in order that appropriate corrective action can be taken. Usually, the inspection applies statistical tools and allows an operator to check his own work; the result greatly depends on the discipline of the operator. The premier inspection, the Source Inspection, is a preventative action to discover the root cause of an error (not a defect which is considered as the product of an error). The detection method of Poka-yoke is based on simple sensors to detect three types of deviations: deviations at work-pieces, at work methods, and at identified parameters. For the work-piece deviations, it uses sensors to distinguish abnormalities in terms of weight, dimension, or shape of a work-piece. The second is the work method deviations, it also uses sensors to detect abnormal matters but in terms of motions. For example, the number of times of a

worker's hand breaks the light beam in reaching for a part or counting the number of spot welds being made on the work-piece, and the fixture clamps will not release unless the correct number of moves has been achieved. The last one is about the checking of identified parameters, it shuts down a process once excess pressure, overheating or excessive torque, etc has been found. **Figure 2–2** shows the concept of Poka-yoke work method deviations. This process consists of two limit switches and two pairs of non-contact electric eyes. The limit switches are used to detect the presence of a work-piece on the work carrier and the side by side alignment of that work-piece while the electric eyes are used to check the orientation and translational location of the work-piece. Once an abnormality has been noted, then the Poka-yoke system invokes an alarm through the ANDON and stops the operation so that operators can fix the error.

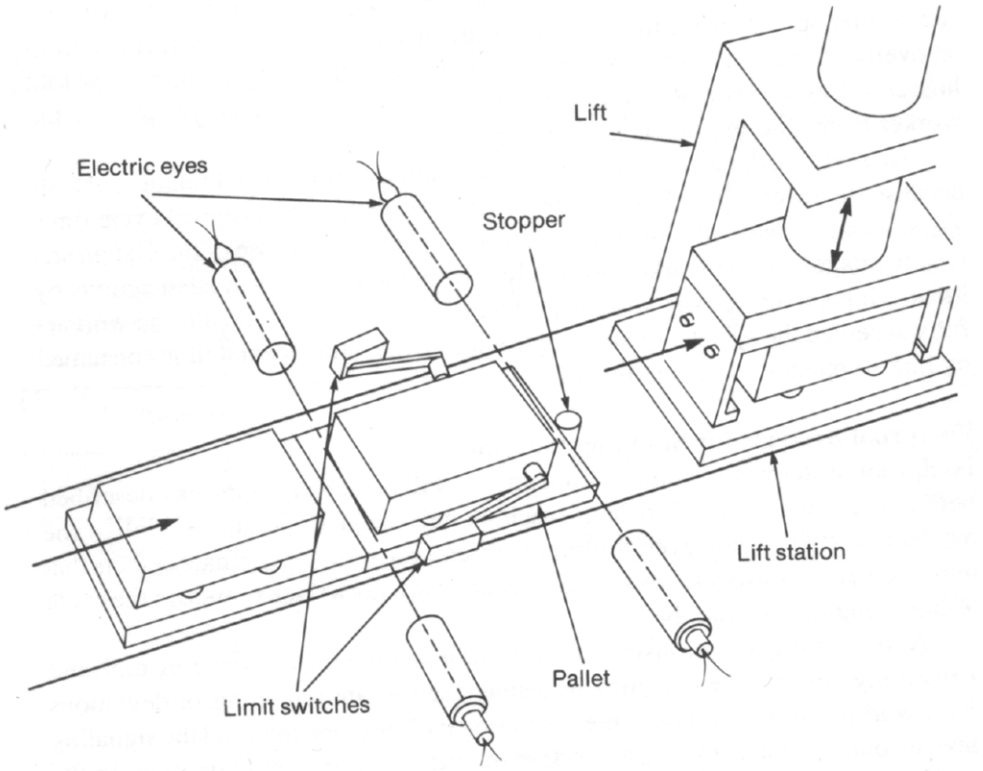


Figure 2-2. Poka-yoke Error Checking (Chen, 1994)

After the review of the Jidoka – Poka-yoke, one can understand that it makes use of simple facilities with the assistance of straightforward reasoning from operators to eliminate defects. Moreover, there is a key feature that is, every operator has the authority to stop the production line which is making or is going to make defective items. This philosophy is more effective in controlling quality and achieves better cost saving than the western Henry Ford's conveyor line concept (Fane *at el.*, 2003), and is also more humanistic. Perhaps, this is one of the reasons that Toyota enterprise has taken over the General Motor and the Ford to become the leader at the field of automotive industry. Yet, the reinforcement of human consistency is still a great challenge in implementing the Jidoka principle.

2.5 Control Chart Pattern Recognition

In 1985, Watanable defined a pattern as the opposite of chaos. It is an entity that can be named. For example, a pattern can be a fingerprint, a handwritten cursive word, a human face, a voice, or a speech signal, etc. The Control Chart is a key SPC member for indicating the process conditions such as the well known Shewhart Chart and the monitored process is said to be out of control when the data collected is not sitting between the two control limits on an x-bar chart (Jia *et al.*, 2001). However, this method may not truly reflect all the characteristics of a process because the trend (e.g., going to good or to bad) is not able to be captured. Therefore, in 1993, Toh and Devanathan (1993; 1994) proposed the recognition of unnatural patterns on a Control Chart by applying the Proportional-Integral-Derivative (PID) control approach so that more information on the production process can be obtained. In most cases, the unnatural patterns come from non-random causes. They have been classified into six

forms of patterns: upward trend, downward trend, upward shift, downward shift, cycle and systematic variation (Guh, 2005).

A system is running normally if it is subject to random causes only when the control line carrying a pattern on a Control Chart is in chaos. Scholars have applied scientific methods (Toh and Devanathan, 1993) to recognize unnatural patterns. Jain *et al.* (2000) categorized the pattern recognition approaches into four branches which are: Template Matching (Yang and Yang, 2005), Statistical Pattern Recognition (Walpole, 1993; Ostle, 1996; Hassan *et al.*, 2003), Syntactic Pattern Recognition (Liu, 1998; Lewis and Ransing, 1997) and Artificial Neural Networks (ANN) (Haykin, 1999; Hwarng and Hubele, 1991).

The Western Electric Company Rules (WECR) was an early method developed to recognize unnatural patterns (Western Electric Company, 1958). Most unnatural patterns can be identified except for systematic unnatural patterns. Such kind of patterns can be identified by some tedious methods as mentioned at the last paragraph. However, the study of dedicated pattern recognition technique is so complex that it warrants a separate research study. WERC is a relatively simple method with a set of rules to guide the necessary reasoning. Recently, the WECR has also been applied in real-time monitoring in an inventory management system and the study showed that the predictable errors of an inventory system can be corrected before an actual failure happened (Cheng and Chou, 2008); the so called natural behaviors also refer to random causes which are somehow non-predictable.

As observed, pattern recognition is a useful technique for distinguishing information embedded on a Control Chart apart from solely observing whether there is any data-point lying outside the two control limits. **Table 2–1** summarizes the approaches applied in pattern recognition and their operational roles.

Table 2-1. The Pattern Recognition Approaches

Pattern Recognition Approaches	**Recognition Technique**
Proportional-Integral-Derivative	Nonlinear regression based on least squares estimation
Template Matching	Correlation, Distance Measure
Statistical Pattern Recognition	Training and Discriminating Function
Syntactic / Structural Pattern Recognition	Decomposition of sub-patterns by Machine learning – Grammar
Artificial Neural Networks	MPL (multilayer perceptions)
WERC	Rules

2.6 Observations on Formulating a System Monitoring Model

Referring to the previous review on system monitoring in this chapter, most of the Western approaches on system modeling were in general dedicated to specific tasks, however, the Japanese ANDON approach seems more flexible but more human intervention can cause inconsistency. Poka-yoke philosophy makes use of simple detecting devices such as mechanical stoppers or Boolean proxy sensors and this can be a good start to generalize the system monitoring architecture. Furthermore, if some common rooted factors (e.g., slowing down and blocking, etc.) can be isolated in the form of patterns in a system, then it is possible for a universal monitoring approach to be formulated to address the monitoring requirements of unlike systems; no matter if it is a traffic system or a production system. The Poke-yoke method identified errors by checking for abnormalities and this aligns with the idea of monitoring patterns on a Control Chart. It must be borne in mind that the finding of an abnormality does not

imply the occurrence of a defeat but it shows an error has been detected. This is because a random cause should not generate a pattern in any form. A pattern is able to tell a lot about how the system is functioning and this can be an important piece of information. This is preferable to waiting for a defect (or a problem) to happen.

To create a simple scientific framework that can be employed for system monitoring generically is worth investigating as this can take advantage of both the Western and the Eastern wisdoms. Normally, establishing dedicated models for different facilities may not be a good method according to the system point of view. Perhaps, monitoring the flow of certain elements can help us to understand a system and this somehow has a strong correlation with the traditional Chinese doctors' practices. The meaning of "simple" is simple in hardware with little variety of modeling for different facilities. Besides, to maintain consistency, an easy way to help the analysis of the information obtained the Poka-yoke philosophy can be very useful. Additionally, the pattern recognition technique can be used to identify the conditions of a system based on the concept of knowing those embedded patterns and the question is, what is the most essential matter to watch? To answer this question, in terms of a dynamics system, something is changing with respect to time and therefore, it is sensible to identify an element type show how it flowing in that system. This will reveal the truth about the condition the system. To summarize, extracting the advantages of both empirical and scientific approaches gives the foundation of this research. Concerning the portability of ANDON and the automation of CIM, it will be good to have a new approach that is highly adoptable and is also able to overview the entire system automatically.

3 Chapter Three – Formulation of Element Flow Reasoning Methodology

The system view may not require in-depth technical details that are essential to the understanding of how a process operates because this viewpoint usually consists of having a macro view of the entire system. In fact, this knowledge is embedded in the flow characteristics of a representative element type and it is anticipated that the operational trend can be reasoned out from examining these sorts of flow in a system and by doing so an observer can understand the performance of the system more easily. The aim of this section is to introduce the way in which the conceptual model can be developed to explain the condition of a system. Subsequently, by examining how the element is flowing, there is a good opportunity of achieving a diagnosis in time to preserve the health of the system.

Methods applied in the field of system monitoring have been discussed in the preceding chapter. These methods can be categorized into either those shaped for a specific system (or type of systems) operating in some unique environments, or those which are flexible but dominated by human interventions. We anticipate that the dynamics system conditions can be inferred by extracting information regarding the element flow. Also, the monitoring setup should be as simple as possible. Moreover, this technique should not be limited to one particular system only and the undesired human intervention should be avoided if it is possible. In this chapter, the proposed system dynamics identification method will be presented in detail.

3.1 Dynamics System Monitoring Perceptions

The capturing of system information starts with looking at a representative element type in the system. The smoothness with which this element flows reflects the healthiness of the system. To keep the configuration simple and giving the data capturing method a good potential for generic applications, time stamping (sometimes called counting) devices are selected and installed to collect information of an element flow stream. With the help of these counting devices, the number of objects (or elements) passing through a check point, which is equipped with a timing device which records the time that the element goes past, can be logged. Then, by means of the coming proposed methodology, it is anticipated that errors – “illnesses” in the system can be diagnosed by analyzing the information from these counting devices. To have an overall picture of the system, we can concentrate on the smoothness of the flowing of the selected representative element type. This concept has good prospects of being applied to different systems.

The condition of an element flow stream, which is the way in which the elements are moving in the section, can be observed by monitoring an element’s “in” and “out” clock time. Knowledge of possible durations being spent at a branch in that network is also needed. **Figure 3–1** shows the schematic diagram of an element flow network of a system using the proposed methodology. To keep the hardware and setup simple, generic hardware is suggested. This is because employing a variety of dedicated monitoring devices will increase the complexity and ultimately, this may lead to the development of dedicated models for different jobs. Therefore, we suggest a straightforward approach in which only the basic time stamping function is needed. The choice of time stamping device is unrestricted as long as it can record the clock

time of each element when it passes the check point. In **Figure 3–1**, there is a pair of counting devices, one is at the beginning and another is at the end of a branch in the network; in other words, each check point stands for one counting device. The flows are supposed to be unidirectional and the section between a pair of counting devices represents a **Region of Interest** (ROI) that must not contain any junction node. Referring to the figure, there are five ROIs in total, each of which lies between two counting devices. Although the dynamics of the entire system network can be complicated, the technique used to examine a ROI can be made simple so that the modeling can lead to an understanding of the ROIs with the intention of gaining knowledge of the entire system.

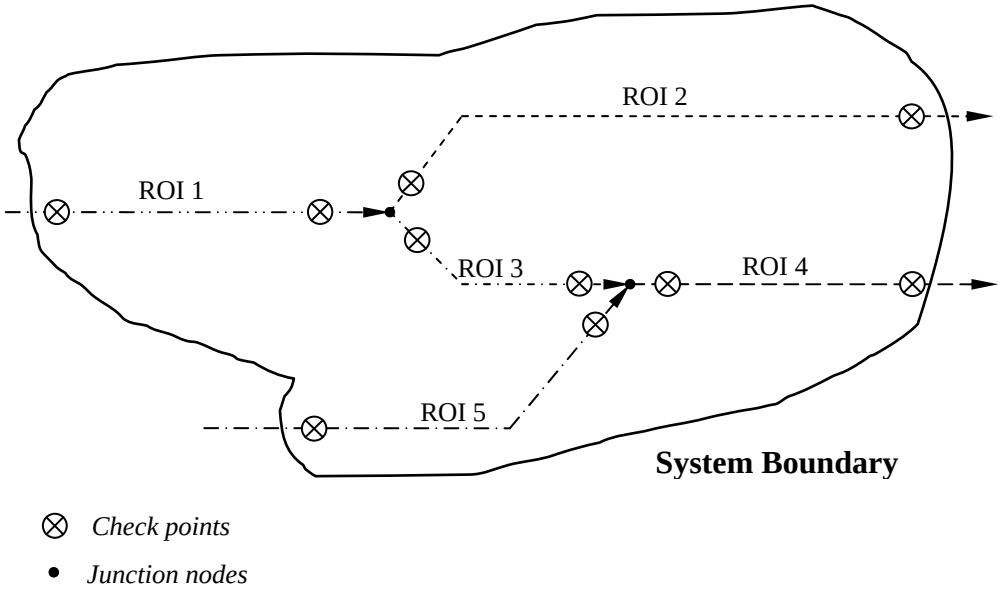


Figure 3-1. System Flow Network

3.2 Notations for Mathematical Modeling

The modeling is concerned with the clock time at which an element arrives at the inlet and then at the outlet of an element flow stream (ROI), together with the knowledge of the normal duration that the element should spend between these two measuring points. This information is sufficient to enable the observer to deduce the condition of the element. The principle idea is that by comparing the differences between the predicted clock time and the measured clock time when an element reaches its outlet, it is possible to perceive the happening of disturbances in that ROI. Since the modeling involves strict timing factors, to steer clear of potential confusion, we use "t" to signify a calculated clock time, "*t*" for a measured clock time, and "T" for a time period in formulating equations. To facilitate the presentation, a set of notations that are used in this research is given in **Table 3–1.**

Table 3-1. Mathematical Modeling Notations

K	A natural number from 1, 2, 3, …, k, …, K to indicate a sequence of known events in the ROI;
β	Element positional shifting index: β-1, β ;
n_k	A positive integer from 0_k, 1_k, 2_k, …, n_k, .., N_k to indicate the positional order of the k^{th} event;
$\boldsymbol{t_{i,\beta}}$	Measured current element arrival time at inlet;
$\boldsymbol{t_{o,\beta}}$	Measured element leaving time at outlet;
$t_{o,\beta}$	Deterministic element leaving time taken at outlet;
$t_{k,\beta}$	Deterministic arrival time at the k^{th} event;
$t_{k,p}$	Initiation time of the k^{th} event;
$t_{k,p,m}$	Initiation discrete time distributions of the k^{th} event with m discrete distributions;
T_g	Minimum time gap between two adjacent elements;
T_s	Ideal throughput time without affect caused by any event;
$T_{k,p}$	Cycle time of the k^{th} event;
$T_{k,pc}$	Effective time on an element of the k^{th} event;
$T_{k,po}$	Non-effective time on an element of the k^{th} event;
T_k	Deterministic time from the inlet to the k^{th} event excluding affects due to previous events;
$\check{T}_{k,\beta}$	Deterministic time from the inlet to the k^{th} event including effects of previous events;
$\Delta T_{k,\beta}$	Deterministic delay time caused by the effective time of the k^{th} event;
T_e	The constant time period for the inter element arrivals at the inlet;
T_x	The calculated repeating cycle of a master signature on the ILT chart.

Before starting to formulate the model, it is necessary to spell out assumptions that will be made here. First, there is no leakage along a ROI so that no element will disappear in a ROI. Second, the effect of an event may cause a time delay on an element, which is comparable to waiting for a gate to open before the element can move on. Third, no element should pass over another element along a flow stream; this is just the popular first-in-first-out (FIFO) rule. Forth, the speed of every element is constant: acceleration and deceleration are both ignored. In addition, with regard to the suggestion of monitoring a system by viewing how elements flow in the network with the concept of identifying ROIs, the modeling method of each ROI could be made alike. Therefore, the modeling can focus on one representative ROI in the following discussion.

3.3 Event Distributions and Event Chain Modeling

To perform monitoring on a ROI, we need to know how it behaves normally so that any oddity can be recognized. Thus, by bringing in the possible dynamic characteristics, known events involved in the ROI should be modeled to simulate the natural behaviors that could happen. To model the event distributions along an element flow steam, such as the k^{th} event to happen in the ROI, the undisturbed[1] time period from the inlet to that event (T_k) needs to be determined. **Figure 3–2** is an example that contains multiple events (1^{st} event, 2^{nd} event, …, k^{th} event, …, K^{th} event). In the figure, undisturbed event time periods from the inlet to each event and the undisturbed throughput time of the ROI (T_s) are shown.

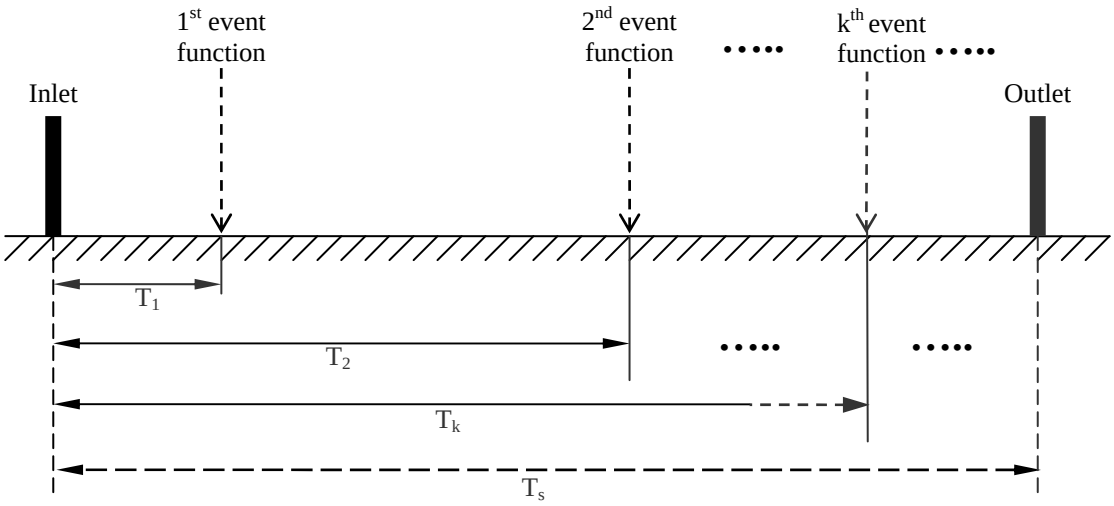


Figure 3-2. Event Distribution in a ROI

Notes
1 - Undisturbed means no time delay on the current element has been imposed at all..

Apart from knowing the distribution of events, the consequences for an element as a result of the event also needs to be quantified. To do this, the time window of an event is modeled as in **Figure 3-3**, in which the effect of the k^{th} event is represented by a pulse chain with the cyclic time interval ($T_{k,p}$) that comprises two sections such that one causes an actual time delay ($T_{k,pc}$) and another specifies the duration gap to the next cycle ($T_{k,po}$); the former is also labeled as “No_Go” on the y-axis since it does cause a time delay if an element hits that zone while the immediate follower is in the “Go” region because it does not induce any effect with respect to time in that element. To synchronize the effect of the event with the clock time, the initiation clock time ($t_{k,p}$) must be known. Each pulse in the pulse chain is tagged in an ascending order (0, 1, …, n_k) in order to trace its position in the clock time window. In case the event is not recurrent, the “Go” duration can be set to a very large value (or to infinity).

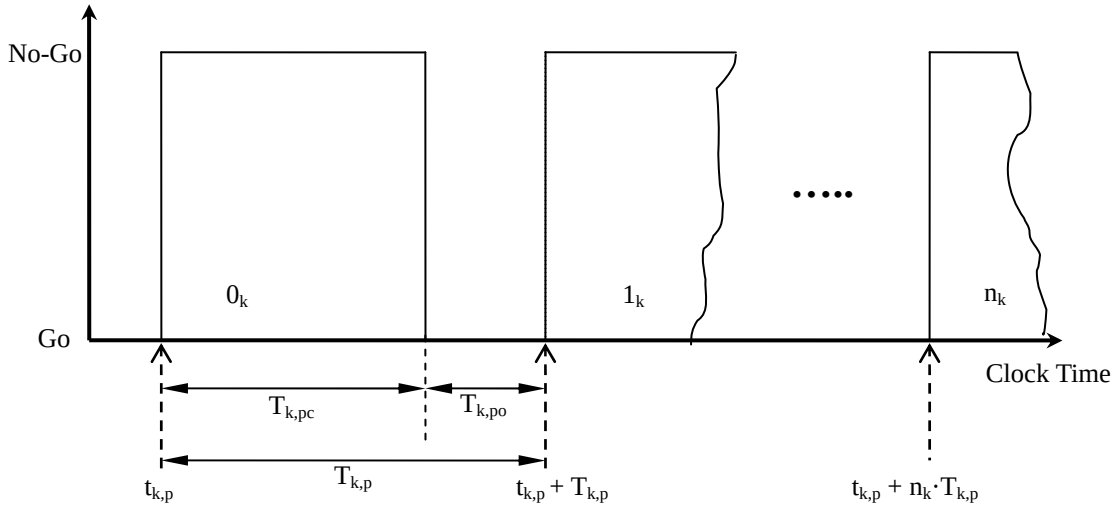


Figure 3-3. The k[th] Event Model

3.4 Element Flow Modeling

First, it is easier to introduce the modeling concept by starting with no hindering caused by the preceding element. Then, the current element arrival time at the k[th] event is affected by the arrival clock time at the inlet plus the time duration required to travel from the inlet to the k[th] event and the cumulative effects from the 1[st] to the (k-1)[th] events. Thus:

$$t_{k,\beta} = t_{i,\beta} + \breve{T}_{k,\beta} \tag{1}$$

To catch the positional order of the event cycle (n_k) of the k[th] event that is going to hit the element, the positional order of the event cycle must fulfill the following condition:

$$t_{k,p} + n_k \cdot T_{k,p} \le t_{k,\beta} < t_{k,p} + (n_k + 1) \cdot T_{k,p} \tag{2}$$

Once the positional order of the hit event cycle has been identified, the possible delay can be calculated. If the arrival time at the event falls into the No_Go period of that cycle ($T_{k,pc}$), then delay is caused and this delay depends on how long that No_Go region remains. Otherwise, there is no effect of it being in the Go region ($T_{k,po}$). The

possible delay ($\Delta T_{k,\beta}$) can be determined by:

$$\Delta T_{k,\beta} = \max[\ (t_{k,p} + n_k \cdot T_{k,p} + T_{k,pc}) - t_{k,\beta}\ ,\ 0\ ] \tag{3}$$

With reference to Equation (1), we need to work out the effects introduced by the 1st to the (k-1)th events. This can be done by using the results obtained from Equation (3) and hence, it can be worked out as:

$$\breve{T}_{k,\beta} = T_k + \sum_{i=1}^{k-1} \Delta T_{i,\beta} \tag{4}$$

If there are totally K events and with the knowledge of the element arrival time at the inlet, then the element leaving time, taking into consideration the delays caused by all events, is:

$$t_{o,\beta} = t_{i,\beta} + T_s + \sum_{i=1}^{K} \Delta T_{i,\beta} \tag{5}$$

Now, it is time to take into consideration the possible influence due to the previous elements. The idea is to look at the immediately preceding element, as the result of the cumulative effects that the immediately preceding element has received should have been embedded in its arrival time at the k^{th} event. If it catches up with the current element, a delay will be imposed. To do this, we need to update the time that the current element arrived at the k^{th} event to see whether it needs to wait for the preceding element to get through that event or not. By checking the new element's arrival time against the preceding element's arrival time plus the minimum time gap in between these two events, the new arrival time of the current element at the k^{th} event can be determined as:

$$t_{k,\beta} = \max[\ (t_{k,\beta-1} + \breve{T}_{k,\beta-1} + T_g)\ ,\ t_{k,\beta}\ ] \tag{6}$$

If the new arrival time is somehow affected by the preceding element, then the positional order of the pulse chain of the k^{th} event that hits the element shifts forward

and it can also be located by applying Equation (2). Then, with the help of Equation (3) to Equation (5), the element's leaving time, by taking into consideration possible hindrance from the previous elements, can also be predicted.

3.5 Initial Event Time Variations and Learning Period

In order to improve the adaptability of the model, some fine tuning on possible variations on the event initial time is imposed. This will enhance the model. As a result, the original initial time of the k^{th} event is modeled by the following discrete distribution function, to replicate the variations:

$$t_{k,p} = f(t_{k,p,m}) \quad \textit{where} \quad \sum_{i=1}^{m} P(t_{k,p,i}) = 1 \tag{7}$$

$P(t_{k,p,i})$ is the probability weighting of each discrete time-slot ($t_{k,p,m}$). The summation of $P(t_{k,p,i})$ for all possible ($t_{k,p,m}$) is equal to 1. Once the event initiation time is represented by discrete distributions, the calculations have to be modified to adopt that change but the calculation logics are just the same. Indeed, the adjustment involves the tracing of each discrete time-slot ($t_{k,p,m}$) with an associated weighting ($P(t_{k,p,m})$) rather than a single event initiation time ($t_{k,p}$) here. Consequently, the forecasting result will also be distributions of different time-slots with associated weightings for each time-slot. For example, if there is a 3 Event Case such that the 1^{st} event has 2 possible time distributions, the 2^{nd} event has 3 and the 3^{rd} event has 2, the calculated output will have 2x3x2 =12 time-slots with 12 associated weightings. **Figure 3-4** is the schematic diagram of the state diagram of this 3 Event Case. One can see that the deterministic leaving time also is a set of discrete distributions ($t_{o,\beta,m}$) with corresponding weightings ($P(t_{o,\ \beta,m})$) for each. In the figure, the whole distribution population is shown and with the knowledge of the actual leaving time of

the immediately preceding element ($t_{o,\beta-1}$), those with values smaller than the preceding element should be eliminated according to the FIFO rule; the last column illustrates the perception of this operation. That means the distributions of the possible leaving time can be adjusted by knowing the leaving time of the preceding element and this is simply governed by:

$$x > t_{o,\beta-1} \quad \text{where} \quad x \in \mathrm{t}_{\mathrm{o},\beta,\mathrm{m}} \tag{8}$$

After the adjustment of the forecasting of possible leaving time of an element, it is necessary to recalculate the weightings and this is achieved by scaling up by the same proportional amount those that remain ($\mathrm{t}_{\mathrm{o},\beta,\mathrm{m}}$). This will normalize the distribution. In effect, this can be done by dividing the individual valid weightings by the summation of all the valid weightings that remain.

$$\mathrm{G}(x) = \frac{\mathrm{P}(x)}{\sum_{x=1}^{\mathrm{x}} \mathrm{P}(x)} \tag{9}$$

where x = 1, …, X is all $\mathrm{t}_{\mathrm{o},\beta,\mathrm{m}}$ satisfied Equation (8).

Indeed, with a better understanding of the ROI being monitored, the number of time-slots will eventually converge to a unique value in view of the fact that the effect is cumulative. The time spent before reaching such a stage can be considered as a learning period. Once a ROI has gone through that period, the event chain concerned starts to work with good synchronization and this provides an important foundation for the model to be finely tuned to the working environment automatically. One minor point that should be mentioned is that there are usually different numbers of events in different ROIs and therefore, their learning periods can also be different.

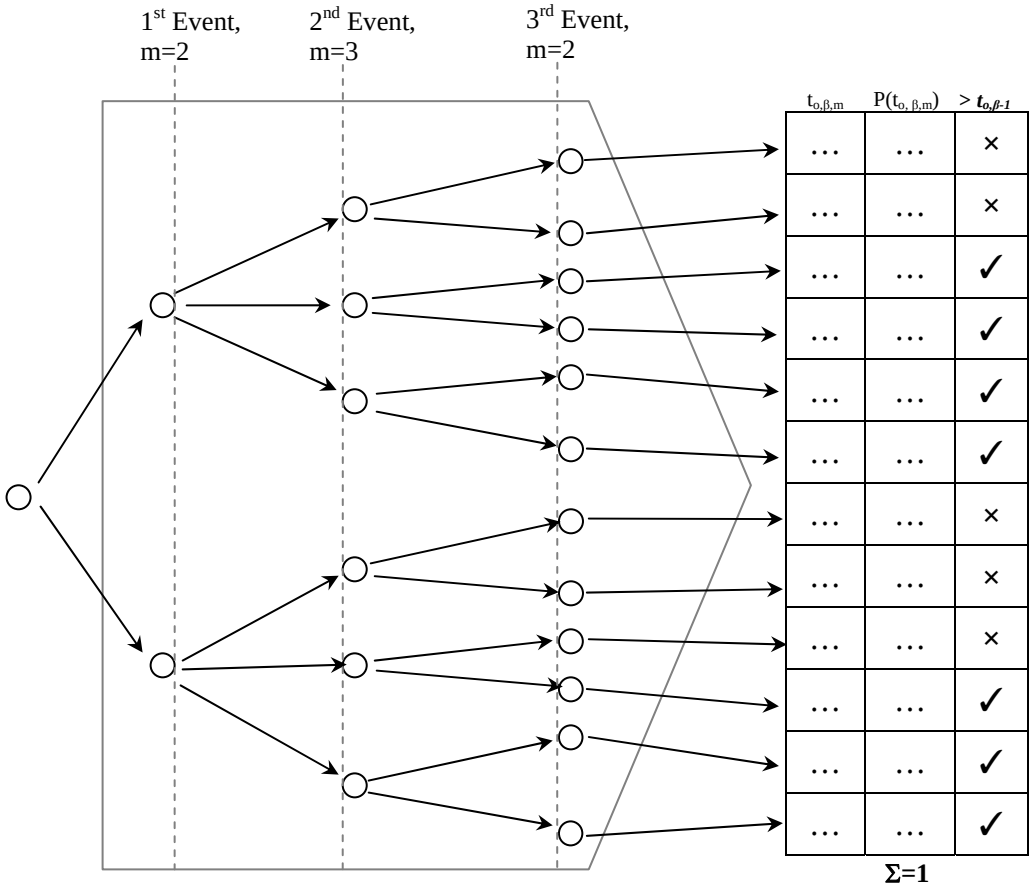


Figure 3-4. The Possibility State Diagram (3 Event Case)

3.6 Regional Index and Inter-element Leaving Time Signature

To analyse the system, an index to reflect the conditions and a deterministic master signature of a ROI have to be established. As discussed before, the predicted leaving time of an element that has just entered the ROI is in the form of discrete distributions initially but it will be gradually converge[2] to a steady value. After the learning period, the deterministic element leaving time ($t_{o,\beta}$) will be a sole value. With reference to the measured element leaving time ($\boldsymbol{t_{o,\beta}}$), the Regional Index (RI) to indicate the conditions of the monitored ROI is defined simply as:

$$RI = t_{o,\beta} - t_{o,\beta} \tag{10}$$

[2] Also see Section 5.1 for the example of handling the learning period.

To diagnose the condition of the system, Regional Index (RI) Charts & Inter-element Leaving Time (ILT) Charts are going to be applied. The RI Chart is constructed by plotting the RI values against the element sequence. **Figure 3–5** illustrates an example of a RI chart. The chart contains a horizontal line across the RI=0; it shows that the ROI is running as expected.

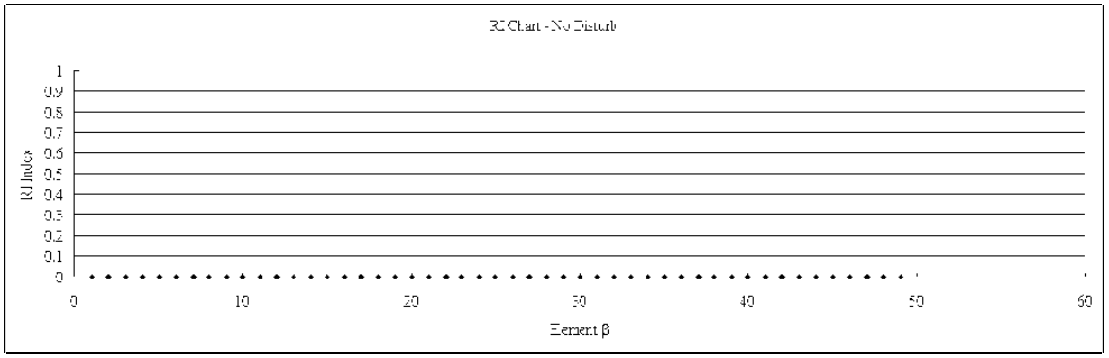


Figure 3-5. The RI Chart

The ILT Chart is constructed by plotting the inter-element time, see Equation (11), of the leaving element against element increments ((β-(β-1)).

$$ILT = t_{o,\beta} - t_{o,(\beta-1)}$$

for the deterministic master signature

or (11)

$$ILT = \boldsymbol{t_{o,\beta} - t_{o,(\beta-1)}}$$

for patterns by measurement

An example of the deterministic master signature is shown in **Figure 3–6** which is a unique signature of a ROI. The cycle period (T_x) of the master signature can be calculated by Equation (12). When a ROI is slipping away from the normal conditions, perhaps due to the presence of errors, the pattern of the unique signature of the ILT Chart may be distorted. By observing the pattern changes of both charts; cases of error can possibly be detected. The details of the method of analysis will be discussed in **Chapter 5**.

$$T_x = \text{LCM}\,[T_e \text{ and } T_{k,p}] \qquad (12)$$

$$K = 1 \text{ to } K$$

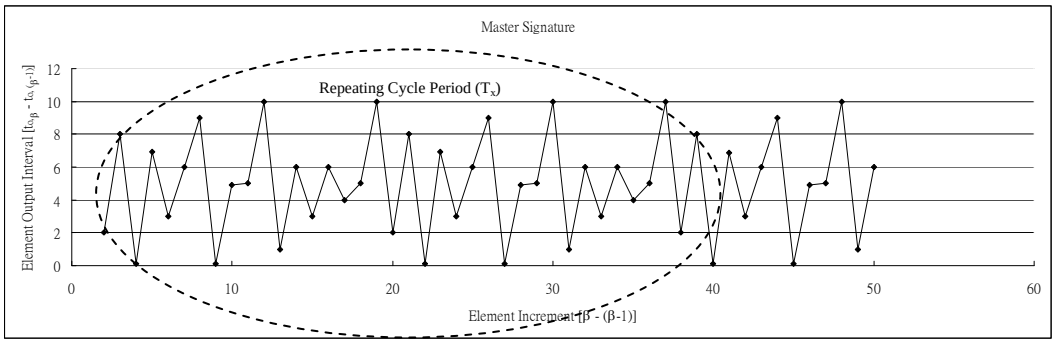


Figure 3-6. The Master Signature – ILT Chart

4 Chapter Four – Element Flow Reasoning Model Software Development

The proposed methodology, the observation of element flows based on the concept of identifying ROIs in association with the developed equation set to monitor the system, was introduced in **Chapter 3**. This chapter describes the software implementation based on the established technique, to determine the required element leaving time. In order to enhance the portability of the software, we have decided to compile a Dynamic Link Library (DLL) to contain all the necessary functions because DLL supports multi-language programs so that programs written in different programming languages can call the functions in the related DLL as long as the programs follow the calling conventions of those functions.

In this research, C++ has been chosen to be the programming language for the required DLL. Then, for the ease of examining results, Microsoft Excel with a macro written in Visual Basic will be employed to call up those functions in the developed DLL. Both the experimental data input and the corresponding program output are handled through the MS Excel worksheets during experiments so that one can view them under a commonly used and familiar environment.

4.1 Element Flow Monitoring Software Architecture

The main purpose of constructing this software program is to verify the mathematical modeling generated in this research. **Figure 4–1** gives an overall picture of the software framework with crucial data connections. The Element Leaving Time Determination (ELTD) module, which is the heart of the software located at the center in the figure, contains the mathematical equation set with the operational sequences established in **Chapter 3**. To determine the element leaving time ($t_{o,\beta}$), basic setups of a ROI including the attachment of those known internal events need to be established beforehand. Then, with the information of the measured element arrival time ($\boldsymbol{t_{i,\beta}}$) and the measured element arrival time of the immediately preceding element ($\boldsymbol{t_{i,\beta-1}}$), the element leaving time can be predicted. To allow the manipulation of multiple I/O tasks, the functions in ELTD would be compiled in form of Dynamic Link Library (DLL).

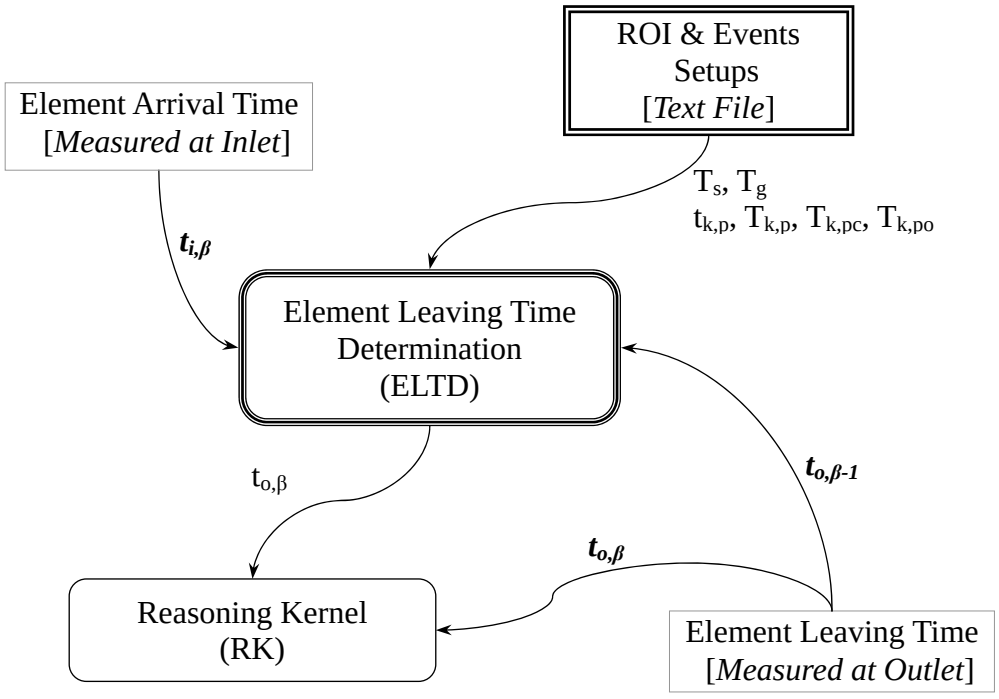


Figure 4-1. The Software Building Framework

4.2 ROI & Events Configurations

It is logical for the information associated with the ROI and the events in it to be stored in a file. The main program will read this information at startup. The suggested file format is the most popular text form (*.txt) with a comma as the separator between two fields and a control character at the end of every line. For ease of reading, a space is inserted between the comma and the next field. The syntax with the ASCII abbreviations in square brackets is:

T_s,[space]T_g[CR]

$t_{1,p,1}$,[space]P($t_{1,p,1}$),[space]…..$t_{1,p,m}$,[space]P($t_{1,p,m}$)[CR]

$t_{1,p}$,[space]$T_{1,p}$,[space]$T_{1,pc}$,[space]$T_{1,po}$[CR]

…..

$t_{k,p,1}$,[space]P($t_{k,p,1}$),[space]…..$t_{k,p,m}$,[space]P($t_{k,p,m}$)[CR]

$t_{k,p}$,[space]$T_{k,p}$,[space]$T_{k,pc}$,[space]$T_{k,po}$[CR]

……

Table 4-1. Example of ROI & Event Cycles Configuration File

380, 1
70, 10, 5, 5
5, 0.20, 6, 0.50, 7, 0.20, 8, 0.1
220, 8, 5, 3
5, 0.30, 6, 0.50, 7, 0.20
330, 12, 8, 4
2, 0.25, 3, 0.50, 4, 0.25

Table 4-1 gives an example of the ROI and events setup file based on the recommended format. In this file, the first line contains the undisturbed throughput time (T_s) and the minimum time gap (T_g) between two elements. This is followed by 3 events. The deterministic time from the inlet to the event ($t_{k,\beta}$), the cycle time ($T_{k,p}$), the effective time on an element ($T_{k,pc}$) and the non-effective time on an element ($T_{k,po}$) are located at the second line and each first sub-line of an event. Then, the second sub-line of each event shows the initial clock time of when that event takes place. In this case, the former one has 4 discrete time-slot distributions in the form of "*time_1, time_1_proportion, time_2, time_2_proportion, ...*" while the following two have 3 possible time-slots for the initiation clock time of the coupled event cycle trains. The unit for the time is unspecified here and one can use it arbitrarily but no decimal point is suggested; with the exception of the weightings where two decimal points are used. In cases where MS Excel is employed, one can call up the DLL functions directly and such a file can be absorbed by specifying a range of cells in the worksheet (or a completely separate worksheet for the purpose of tidiness) to accommodate the required data. Also see **Tables C-1**, **C-2** and **C-3** at **Appendix-C** for the MS Excel worksheets being used for a ROI with its setup data, in order to examine the proposed methodology.

4.3 Element Arrival and Leaving Time Measurements

To keep the hardware setup simple, each ROI has two measurement points only; one is located at the inlet and another is at the outlet. In practice, two simple digital inputs can get the job done but counting devices can be better if the counting frequency is a concern; perhaps, understanding the elements accumulated inside the ROI could also be a piece of useful information – although the current model is not equipped with that function. In fact, what we are interested in, is the timestamp of each element at its arrival and departure. This can be easily obtained by associating the rising edge of a simple signal to the "get current time" function (or something similar) and then recording it as the required timestamp. An example program written in LabVIEW for getting the timestamp when the counter detects a signal has been given in **Figures D-1** and **D-2** at **Appendix-D** for reference. In order to experiment using the model, the arrival time have been input through MS Excel to simulate the element arrival time. One can see the ($t_{i,\beta}$) in row 5 in **Figure D-4** in **Appendix-D**.

4.4 Implementations of the Element Leaving Time Determination Core

With reference to **Figure 4-1**, it can be seen that the implementation of the Element Leaving Time Determination (ELTD) core is the most essential part of the program. Fundamentally, there are two sets of functions: the set of basic functions and the set of enhanced functions for handling the probability operations.The latter also shares some of the basic functions. Normally, the basic functions are used when the learning period has been completed, or when there is no need for such a learning exercise, in cases where the event initiation time is known for sure. The ELTD contains seven DLL functions: model_init, new_event, new_element, model_run( ), get_exp_out_time,

model_reset and model_end. A sequence diagram regarding the operations of these seven DLL functions is shown in **Figure 4-2**. The programming highlights, corresponding to the seven DLL functions, are enclosed in retangular boxes with solid line in the coming discussion. If there are discrete distributions on event initiation time involved that means the application has to go through a learning period and some extra coding work has to be done. To cater for this situation, eight additional functions have been introduced which are: case_model_init, set_case_event_size, set_case_event, build_case_possibility_run, get_case_out_time_size, get_case_out_poss, get_case_out_time, set_case_act_out_time; to differentiate them from the basic functions, they are surrounded by rectanglar boxes with dotted lines, in the following sections. The operational sequence is shown in **Figure 4-3**.

Finally, the parameter declarations are somewhat self-explanatory since they are aligned to the equations constructed in **Chapter 3**. In the following sections, only essential issues will be addressed and the skeleton of each function will be presented. The programme source codes of these functions are provided in **Appendix-A** for reference. Corresponding symbols used in **Chapter 3** are shown in round brackets in the program skeleton boxes.

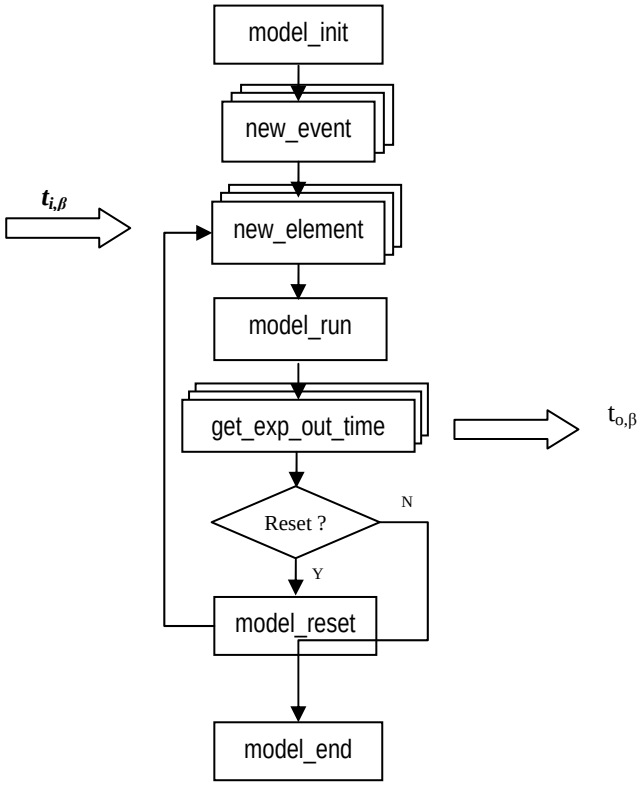


Figure 4-2. ELTD DLL Sequence Diagram – Basic Model

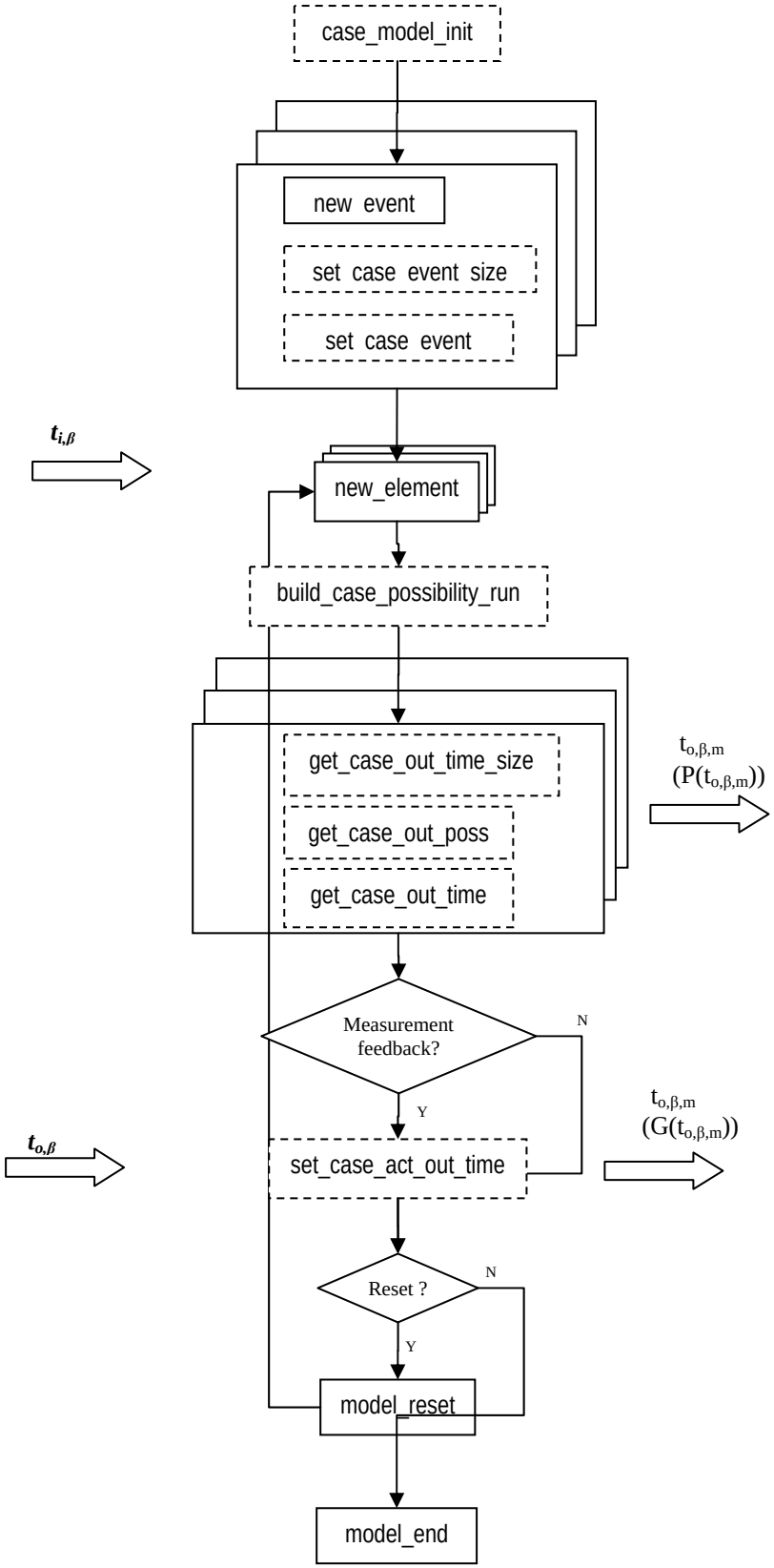


Figure 4-3. ELTD DLL Sequence Diagram – Discrete Distribution Model

4.4.1 The Initialization Functions

The model_init is for starting the program with basic setups such as the throughput time without any events involved (time_standard = T_s) and with the mininum time gap between two adjacent elements (time_gap = T_g). The capacity_element is introduced for the convenience of testing the model testing so as to reduce the number of times that looping is needed since we intend to use MS Excel to perform the testing. The capacity_event is used to tell the total number of events in this execution. If the discrete distributions are involved, then the case_model_init is needed to allocate extra memory for the address pointers.

```
int model_init(double time_standard, double time_gap, int capacity_element, int capacity_event);
     /** initialize process and prepare memory
     @param:
                time_standard                    (Ts) Ideal time to travel from Inlet to Outlet
                time_gap                         (Tg) Min gap between two elements
                capacity_element                 No. of elements in this run
                capacity_event                   No. of events in this run
     @return
                1:                               OK
                -1:                              Failed
     */
```

```
int case_model_init();
     /** init extra memory for propability model for events based on event capacity
     @param:
          n/a
     @return
                1:                               OK
                -1:                              Failed (out of memory)
     */
```

4.4.2 The Event Setup Functions

The basic new_event function responsible for establishing the cycle time of a known event includes the effective time ($T_{k,pc}$), the non-effective time ($T_{k,po}$), the net time to event (T_k), and the actual initiation time ($t_{k,p}$). This function is called upon when an event is going to be added and therefore, it should be executed as many times as the total number of events (capacity_event) that are expected to be present in the ROI. Similarly, if discrete distribuions are required to be drawn in then two more functions are required. The first is the set_case_event_size that is reponsible for allocating memory space according to the number of distributions needed (m) in connection to the event identification (event_id) for each event. The second function, set_case_event_weight, is for seting up the weightings for every event distribution (weight) in a way such that each distribution slot is also assigned an index (weight_id) for the associated event_id for the purpose of tracing the individual weightings.

```
int new_event(double non_eff_time_per_period, double eff_time_per_period, double phase_delay,
double net_time_period);
    /** setup event chain
    @param
         non_eff_time_per_period:   (T_{k,po})
         eff_time_per_period:       (T_{k,pc})
         phase_delay:               (t_{k,p})
         net_time_period:           (T_k)
    @return
              x:                              event_id (range: 0 to max_event_num)
              -1:                             Failed (out of bound) or (events have been set)
    */
```

```
int set_case_event_weight(int event_id, int weight_id, double tm, double weight);
     /** setup distribution values for events
     @param:
               event_id                       event_index
               weight_id                      weight index
               tm                             (t_{k,β,m}) clock time of first event pulse
               weight                         (P(t_{k,β,m})) probability weight
     @return
               1:                             OK
               -1:                            Failed
     */


int set_case_event_size(int event_id, int size);
     /** setup memory size for propable events
     @param:
               event_id                       event_index
               size                           (m) number of weights of the associated event_id
       @return
               1:                             OK
               -1:                            Failed
     */
```

4.4.3 The Element Arrival and Model Run Functions

When an element arrives at the inlet of a ROI, it is necessary to create a timestamp for that element and this function assigns an identification (element_id) with a timestamp (coming_time). In fact, this function also caters for batch calculations with a known data set of element arrival time for experimenting with the model. When monitoring flowing elements, only the two element_ids will be needed which are: element_id = 0 for the previous element and element_id = 1 for the current element. Then, the model_run is called on to perform the calculations as discussed in **Chapter 3.** This

function also takes into consideration the effect due to the previous element ($t_{o,\beta-1}$). In terms of the discrete distributions, additional memory space is needed to store the possible leaving time which have been worked out and this is done by the build_case_possibility_run that replaces the primary model_run function.

```
int new_element(double coming_time);
      /** found a newly coming element at the ROI inlet and store the previous element timestamp
          (=0 if there is no previous element)
    @param:
               coming_time                    (t_{i,β}) element timestamp at inlet
    @return
               x:                             the element_id  (range: 0 to max_element_num for
                                              batch calculations) or
                                              (x=1 for current element with x=0 for previous
                                              element)
               -1:                            Failed (out of bound)
    */
```

```
       int build_case_possibility_run( );
    /** allocate memory for all events combinations and run the model
    @param:
         n/a
    @return
               1:                             OK
               -1:                            Failed
       */
```

```
int model_run( );
     /** run the model to calculate the element leaving time
       @param:
         n/a
     @return
               1:                             OK
               -1:                            Failed
     */
```

4.4.4 The Element Leaving Time Enquiry Functions

The fundamental get_exp_out_time function is used to enquire about the expected element leaving time ($t_{o,\beta}$) at the ROI outlet with reference to the element_id. If we compare the various distributions, the discrete distribution is a bit complicated as there are four functions to handle this situation which are: get_case_out_time_size, get_case_out_poss, get_case_out_time, set_case_act_out_time. The get_case_out_time_size is for determining the total number of possible element leaving time ($t_{o,\beta}$) of the specified element_id required at the outlet. This function is followed by the get_case_out_poss that returns the calculated weightings ($P(t_{o,\beta,m})$) according to the weight_id. The get_case_out_time is used for the purpose of obtaining the leaving time ($t_{o,\beta,m}$) of the concerned element_id with its associated weight_id. The last function set_case_act_out_time is employed to sort out the leaving time of that element in such a way that each element is larger than the immediately preceeding element($t_{o,\beta-1}$).

```
double get_exp_out_time(int element_id);
     /** return an element's leaving time
     @param:
               element_id:                    index of elements (0,1,..., n-1)
     @return:
               x.xx:                          (t_o,β)
               -1:                            Failed
     */
```

```
int get_case_out_time_size(int element_id);
     /** return number of distributions of a given element
     @param:
               element_id                          element index
     @return
               x:                                  (size of t_{o,β,m} at outlet) >=0 how many possible
                                                   cases are available for the element
               -1:                                 Failed
     */

double get_case_out_poss(int weight_id);
     /** return weight of the given combination (by index)
     @param:
               weight_id                           weight index
     @return
               x:                                  (P(t_{o,β,m})) >=0
               -1:                                 Failed
     */

double get_case_out_time (int element_id, int weight_id);
     /** return the leaving of a given element in a given combination case
     @param:
               element_id                          element index
               weight_id                           weight index
     @return
               x:                                  (t_{o,β,m})  >=0 leaving time
               -1:                                 Failed
     */

int set_case_act_out_time(int element_id, double to);
        /** write actual out time back, to re-calculate combinations and their possibility
     @param:
               element_id                          element index
               to                                  (t_{o,β,m})  leaving time subject to > t_{o,β-1}
        @return
               1:                                  OK
               -1:                                 Failed
     */
```

4.4.5 The Model Reset and Model End Function

These are the last two functions. The model_reset should be executed if it is necessary to perform a re-calculation based on the current scenario. This function fills the specified memory with “-1”. Lastly, the final function, model_end, is responsible for releasing all the allocated memory space being used in the programme before the end of the execution. This is a typical practice for every programme.

A word about the use of functions in DLL. Great care should be taken with the choosing of the argument types specially if different programming languages are going to be used. A typical example is that the declaration of “int” (integer) to hold the required range of values can be either 16 bits (e.g., VB 6.0) or 32 bits (e.g., C#) with different programming platforms.

```
void model_reset();
    /** For re-calculation of times for all elements

    */
```

```
int model_end();
    /** free the memory
    @return
         0:              OK
    */
```

5 Chapter Five – Verification and Test of Element Flow Reasoning Model

In this chapter the proposed model will be verified and it will be confirmed that it works as expected. Besides this, its ability to identify various system errors will also be examined. An evaluation of the computational accuracy of the software program developed in **Chapter 4** has been conducted by "Hand Simulation" and the results are given in **Appendix-B**. As discussed, the performance of an ROI is analyzed using signature identification. The details of this are going to be discussed in this chapter.

This chapter starts by introducing the learning function which is employed to handle uncertainties in events. Then, conditions of an ROI will be examined. For example, the ROI is in an abnormal state if any unexpected signature is detected. For experimental purposes, symptoms including Overflow, Slowing down, Blocking or Unknown Event Occurrence will be studied. Before conducting the analysis, the configuration of the ROI must be overflow free. Otherwise, the queue of elements will reach the upstream event and this is a sign of inadequate buffer space.

5.1 The Learning Period

The learning period is equipped to clear uncertainties in events. There are occasions that it is much easier to estimate an event cycle initiation time rather than aiming at exact accuracy. Therefore, a set of discrete time-slots with weightings has been introduced to represent such uncertainty at an event. When there are uncertainties at the beginning time, a set of time in the form of discrete distribution functions can be used. An example that contains 3 events such that the first and the second events have

two possible initiation time while the third one has three possible initiation time is detailed in **Table C-4** in **Appendix-C**; it contains 12 (2x2x3) deterministic element leaving time combinations. The results are shown in **Figure 5–1,** in which within the twelve possible leaving time, the most possible one has a chance of 57%.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	C
1															
2	step 0: clear table			step1: run without observation					step2: read t.actual, and do possibility pruning						
3															
4	Measured current element arrival time at inlet		e1	e2	e3	e4	e5	e6	e7	e8	e9	e10	e11	e12	e13
5	$(t_{i,\beta})$		5	10	15										
6															
7	Measured element leaving time at outlet		e1	e2	e3	e4	e5	e6	e7	e8	e9	e10	e11	e12	e13
8	$(t_{o,\beta})$														
9															
10	∑p=	1													
11	Deterministic element leaving time taken at outlet $(t_{o,\beta})$ possibility		e1	e2	e3	e4	e5	e6	e7	e8	e9	e10	e11	e12	e13
12	abs	relative(%)													
13	0.57	57	21	28	31										
14	0.03	3	21	22	28										
15	0.19	19	18.5	27.5	28.5										
16	0.01	1	18.5	21	27.5										
17	0.106875	10.6875	20	30	31										
18	0.005625	0.5625	20	21	30										
19	0.035625	3.5625	20	27.5	30										
20	0.001875	0.1875	20	21	27.5										
21	0.035625	3.5625	19	28	29										
22	0.001875	0.1875	19	21.5	28										
23	0.011875	1.1875	18.5	27.5	28.5										
24	0.000625	0.0625	18.5	21.5	27.5										

Figure 5-1. Example of Discrete Distribution Results

Although by computation is possible to work out all the possible element leaving time, they may not all occur because it will never be earlier than the immediately preceding element. Therefore, data smaller than the immediately preceding element ($t_{o,\beta-1}$) can be removed. **Figure 5–2** gives the program running results where the upper dotted line rectangular box outlines the effect of knowing the first element's leaving time and the another dotted line rectangular box contains the results after knowing the second element. In the figure, the measured leaving time of the first element is 20 time units

and this reduces the original 12 leaving time combinations to 4. This will be further reduced to a single value once the second element's leaving time (30 time unit here) has been obtained. Studies show that the convergence goes quite fast and a unique predicted element leaving time will be obtained soon after this; this is also the end of the learning period.

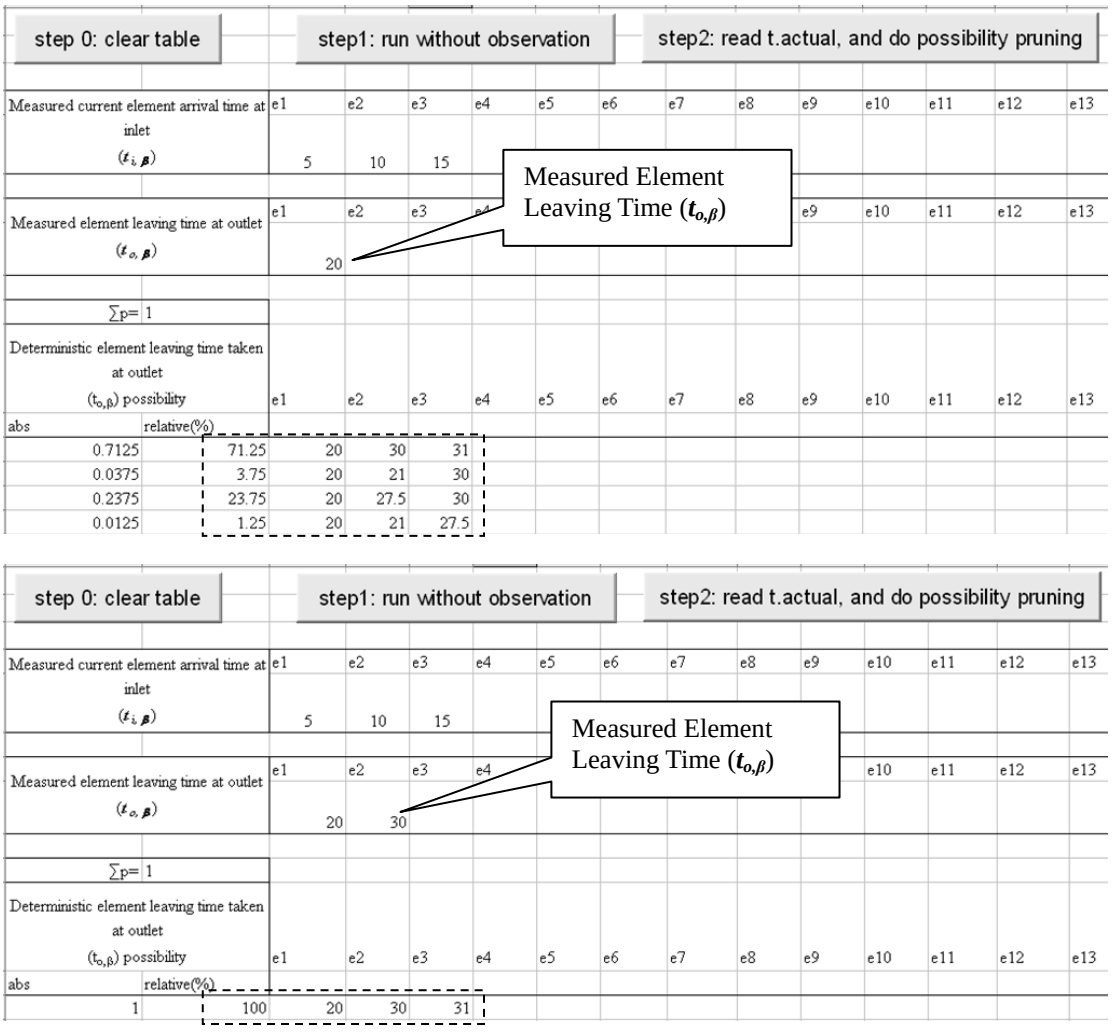


Figure 5-2. Results with Outlet Feedbacks

5.2 Experimental Setup

A set of conditions has been defined for experimental purposes, in which the ROI contains three events, **Figure 5–3** and **Table 5–1** show the basic configurations of the three events involved. Initially, a Normal situation will be used to confirm the operational conformity of the program. Data for this test is given in **Table C-5** in **Appendix-C**. The RI chart has also been constructed in **Figure 5–4** and one can see that all dots lie along the x-axis there. This means after it has passed through the ROI, the deterministic element leaving time ($t_{o,\beta}$) is the same as the actual leaving time ($\boldsymbol{t_{o,\beta}}$) based on the simulated data; and this is what we expected.

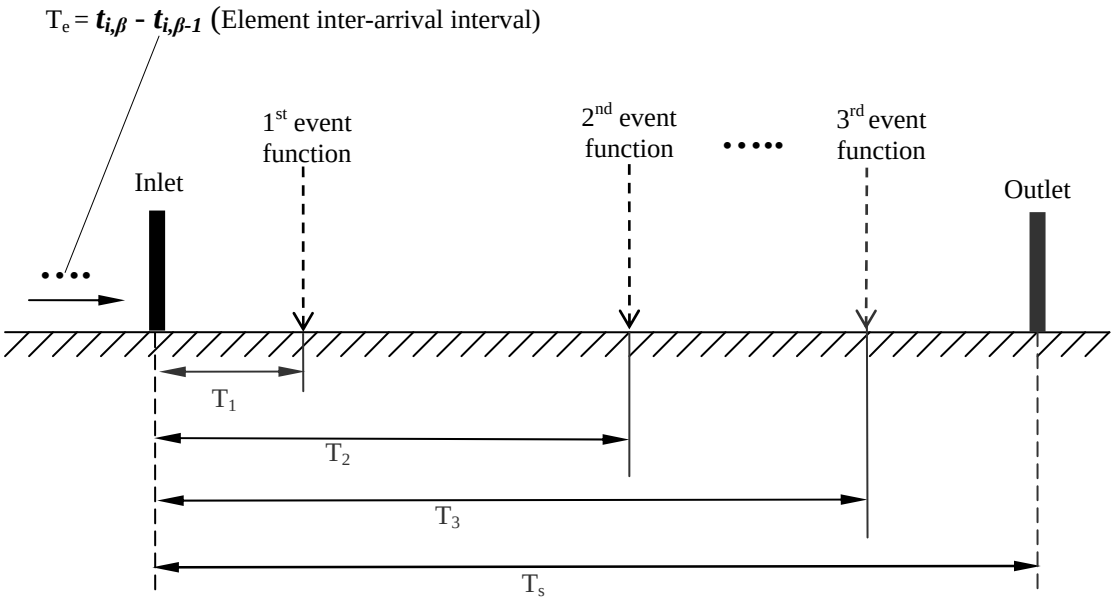


Figure 5-3. Schematic Diagram of Experimental ROI

Table 5-1. Event Configurations (3 events)

k	$T_{k,p}$	$T_{k,pc}$	$T_{k,po}$	T_k	$t_{k,p}$
1	6	4	2	20	0
2	9	5	4	30	0
3	5	2	3	50	0

$T_g = 0.1$, and $T_s = 60$

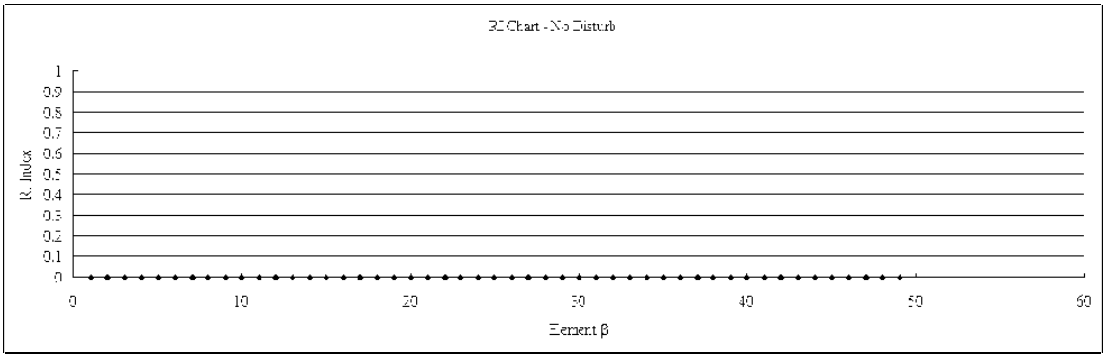


Figure 5-4. Normal Case – RI Chart

The pattern of the ILT chart is repeated. This is considered as the master signature of this ROI. The cycle duration (T_x) can be calculated by taking the LCM of T_e and all $T_{k,p}$ involved, i.e., T_x = LCM(5,6,9,5) = 90 time unit/cycle. Since (T_e) is 5 unit time/element, this pattern will repeat cyclically in 90/5 = 18 element/cycle. The master signature that represents this ROI is shown in **Figure 5–5**, where the maximum amplitude is 10 time units and a horizontal thickened line has been drawn on the graph. Indeed, this value will be used as a reference to signify the trend to error as no time difference between two adjacent elements should be greater than this value. But further analysis will be needed to indentify the error type.

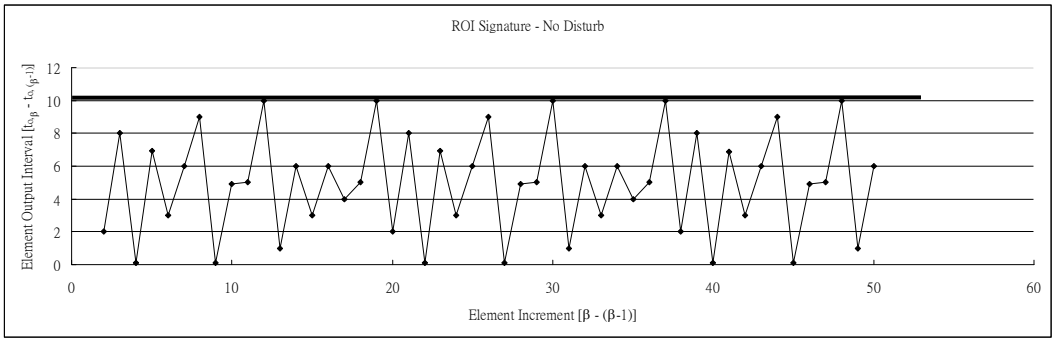


Figure 5-5. Normal Case - ILT Chart

Normally, the Overflow condition should not occur and this can be detected by checking the cycle duration on an ILT Chart; it should be noted that the cycle duration will be shorter than that of the master signature (T_x) if an Overflow occurs. In other words, a signature with duration smaller than (T_x) stands for the occurrence of Overflow.

If (T_g) = 4 time units for the given example, Overflow will occur. In **Figure 5–6**, one can see that the cycle period is smaller than (T_x) and the trend of RI values is increasing in **Figure 5–7**. More information is provided in **Table C-6** in **Appendix-C**.

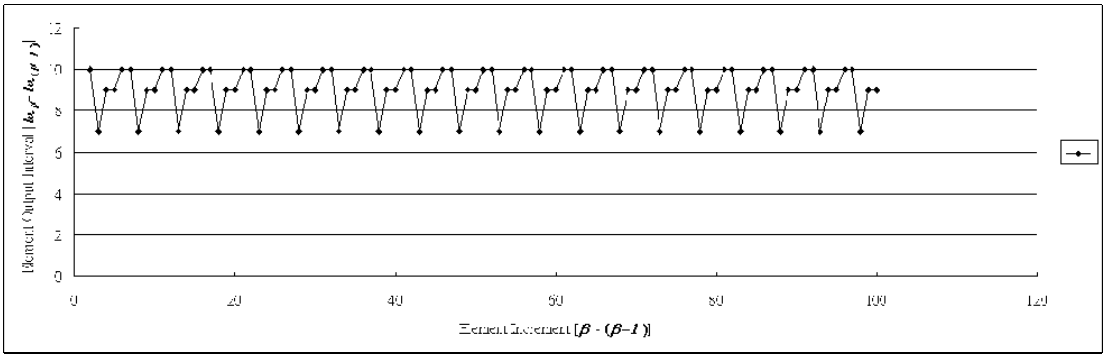


Figure 5-6. Overflow Case– ILT Chart

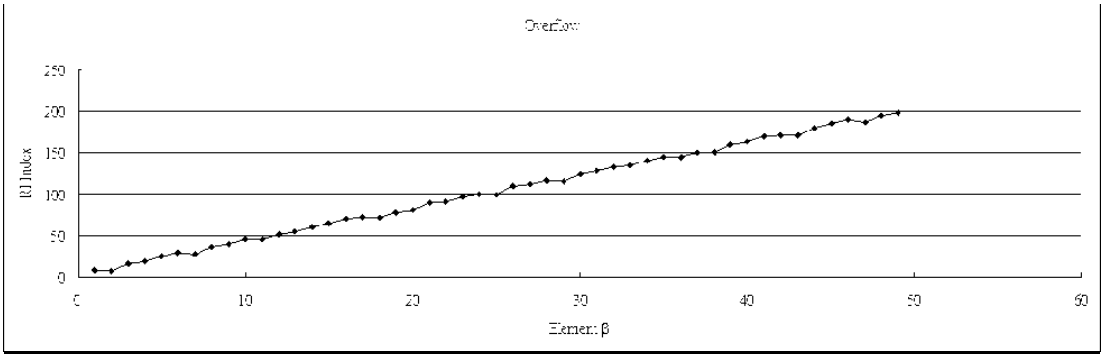


Figure 5-7. Overflow Case– RI Chart

5.3 ROI Disturbance Cases

As one can observe, the two charts are being used to monitor a ROI whereas the RI chart is used to check the deviation from the expected value one by one while the ILT chart is for watching the signature. This section examines the information embedded in these two charts and the nature of their relationship. To understand the relationships, typical problematic cases have been identified to be experimented with. Generally, the resource status of an event (i.e. a workstation of a production line) can be classified into: Active, Idle, Busy, Inactive, and Fail. The last four status can caused the four abnormal conditions in a system (Higgins, 1990; Kelton *et al.*, 2004). The five cases are shown in **Table 5–2**. In fact, the former two cases, Normal and Overflow, have been discussed in the last section. **Slowdown** stands for a kind of disturbance that causes longer traveling time than Normal. **Blocking** means the moving stops[3] for a while at certain points and then continues; this phenomenon can be simulated by adding a "one-off" event. The last is the Unknown Event Occurrence; it is about the introduction of a new event that is not observed by the proposed monitoring model.

Table 5-2. Selected Cases in ROI

Case	Description
1	Normal
2	Overflow
3	Slowdown
4	Blocking
5	Unknown Event Occurrence

[3] Elements are flowing through ROI from inlet to outlet. 'The moving stops' means an element stops somewhere along the ROI for a while and then resumes.

5.3.1 Slowdown in ROI

A Slowdown means the travel time is prolonged somehow; this can be due to a fall in moving speed. To experiment with this condition, four Slowdowns with 3 time units each were setup individually in the ROI; they are: "between Inlet and 1st Event", "between 1st Event and 2nd Event", "between 2nd Event and 3rd Event", between "3rd Event and Outlet". Configuration details can be found in **Table C–7** in **Appendix-C**. The numerical results are given in **Tables C–8** to **C–11** in **Appendix-C**.

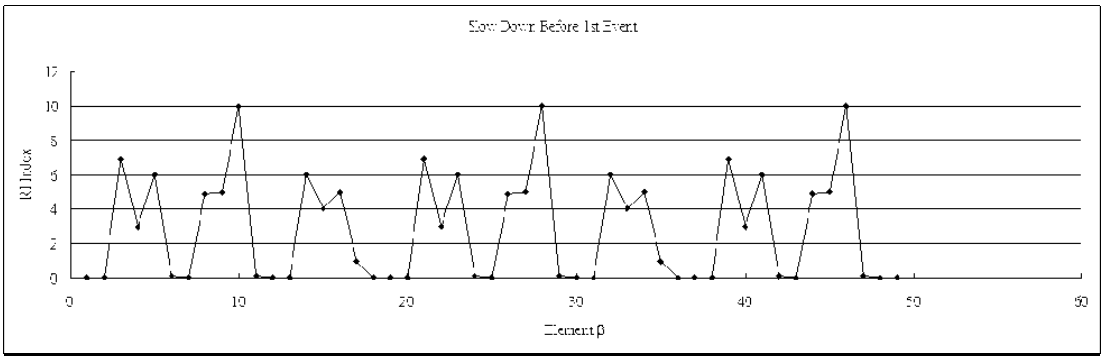


Figure 5-8. Slowdown between Inlet and 1st Event – RI Chart

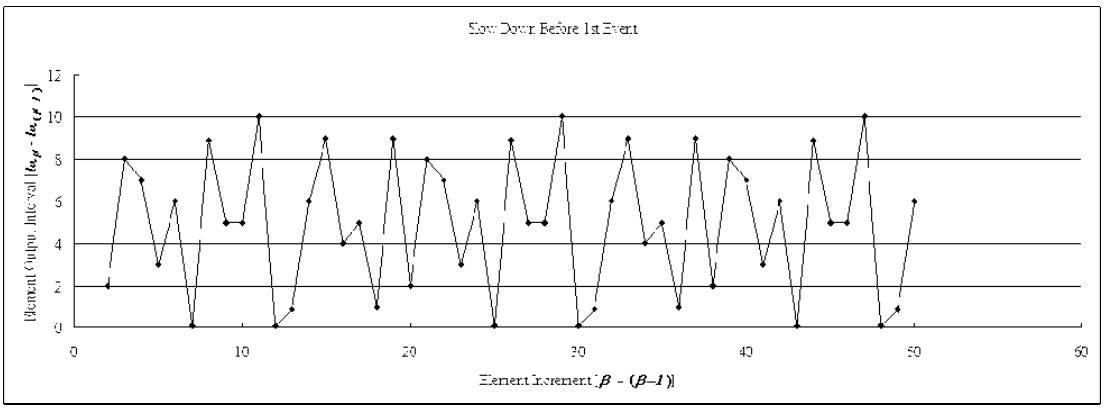


Figure 5-9. Slowdown between Inlet and 1st Event – ILT Chart

The results were plotted in **Figures 5–8** to **5–15**. It can be seen that there is no longer any straight line across zero on RI charts except when the Slowdown takes place after the last event, then there are some repeated cyclic patterns with durations identical to the master signature.

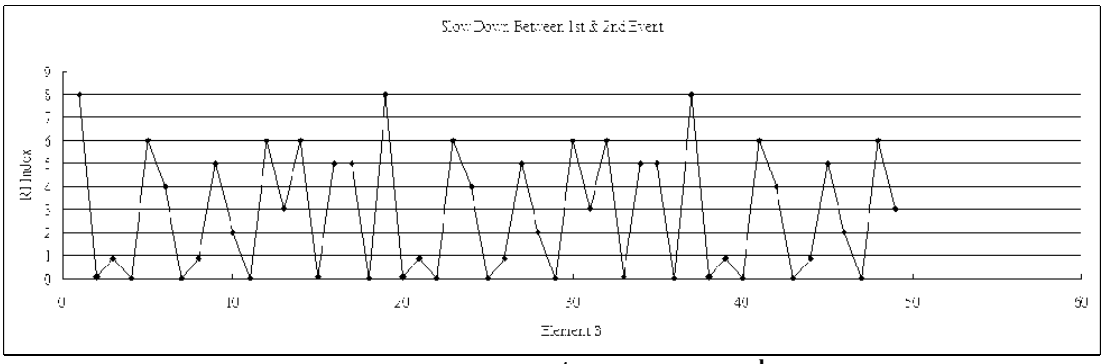


Figure 5-10. Slowdown between 1st Event and 2nd Event – RI Chart

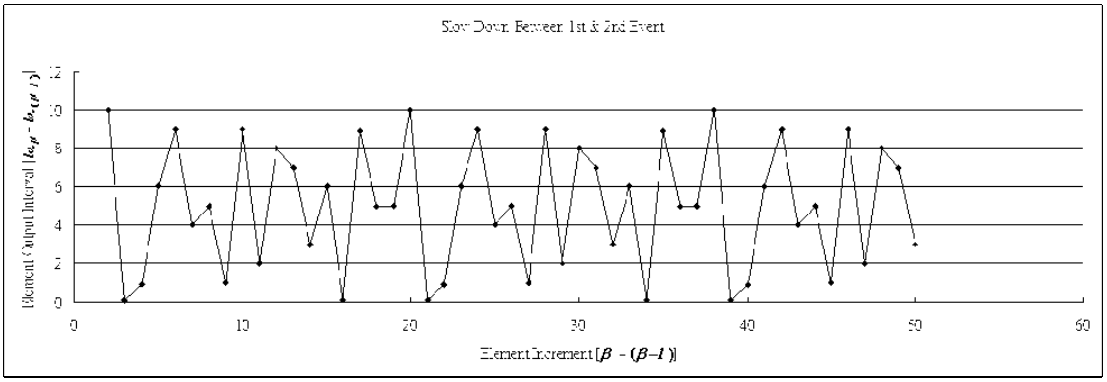


Figure 5-11. Slowdown between 1st Event and 2nd Event – ILT Chart

In **Figure 5–12**, all are above zero and this will only be obtained during a Slowdown condition. When the Slowdown is after the last event, RI chart becomes a straight line with a constant value equal to the actual slowdown time; see **Figure 5–14**.

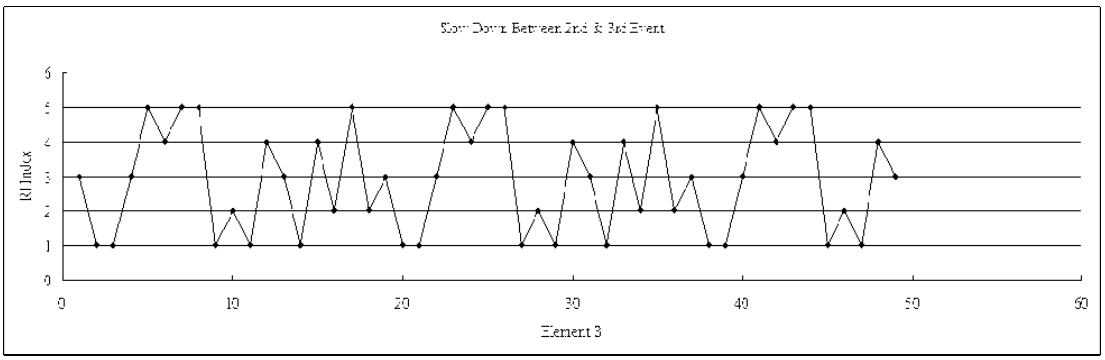


Figure 5-12. Slowdown between 2nd Event and 3rd Event – RI Chart

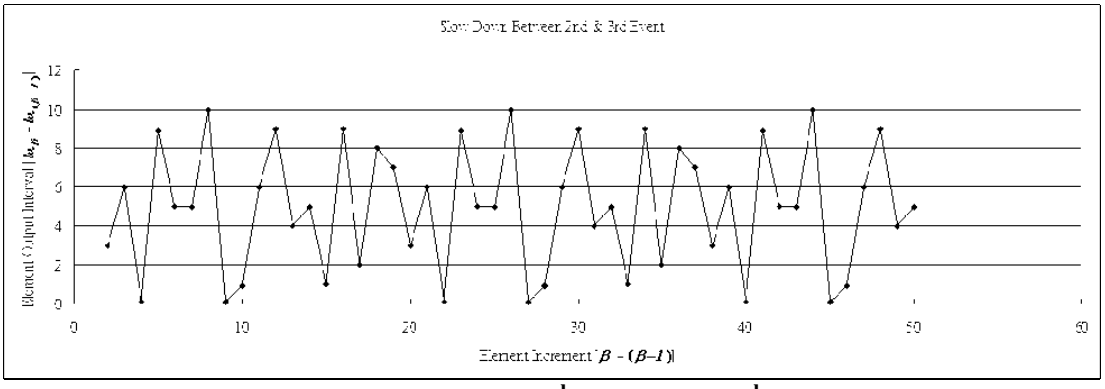


Figure 5-13. Slowdown between 2nd Event and 3rd Event – ILT Chart

In terms of the ILT charts, with the exception of **Figure 5–15**, repeated patterns were obtained. They are dissimilar to the master signature, but the cycle time are the same. **Figure 5–15** presents a Slowdown that happened after the third event (last event) and it is the same as the master signature. In fact, the ROI can be running abnormally even when a master signature is shown on an ILT chart.

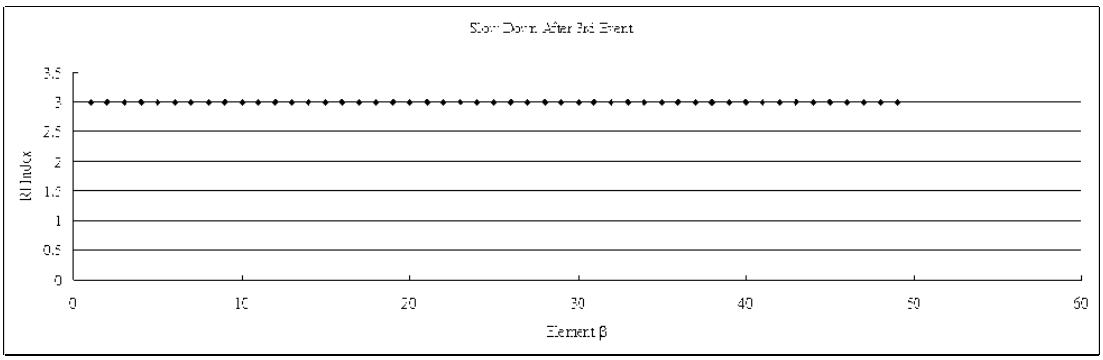


Figure 5-14. Slowdown between 3rd Event and Outlet– RI Chart

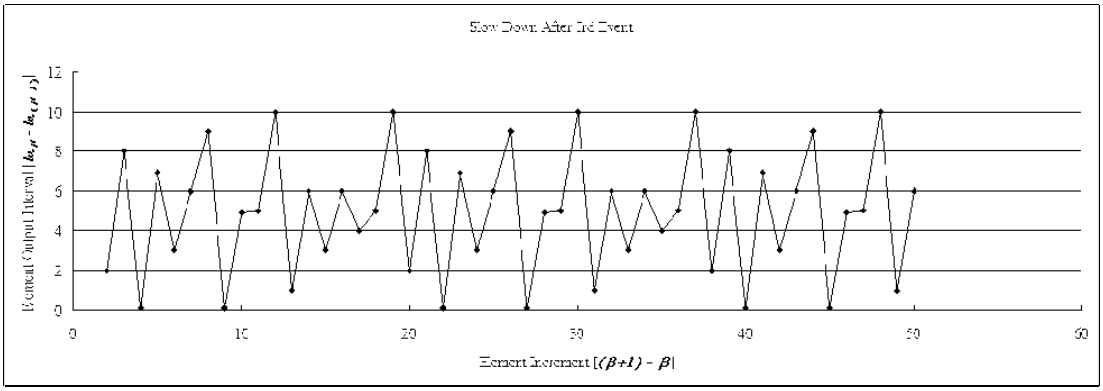


Figure 5-15. Slowdown between 3rd Event and Outlet – ILT Chart

5.3.2 Blocking in ROI

Blocking stands for a stop at a certain position followed by a release. During the experiment, the Blocking is replicated by setting up a one-off event. The configuration details are given in **Table C–12** in **Appendix-C.** Similar to the Slowdown situation, four separate locations were examined.

The blocking duration was for 40 time units and the time it occurred was at the 80 (350-270) time unit. It is observed that when a Blocking comes into effect, a sharp ramp on both RI and ILT charts is found. The experimental results are provided in **Tables C–13** to **C–16** in **Appendix-C**. When a RI chart goes to zero again, the ROI is back to a normal condition.

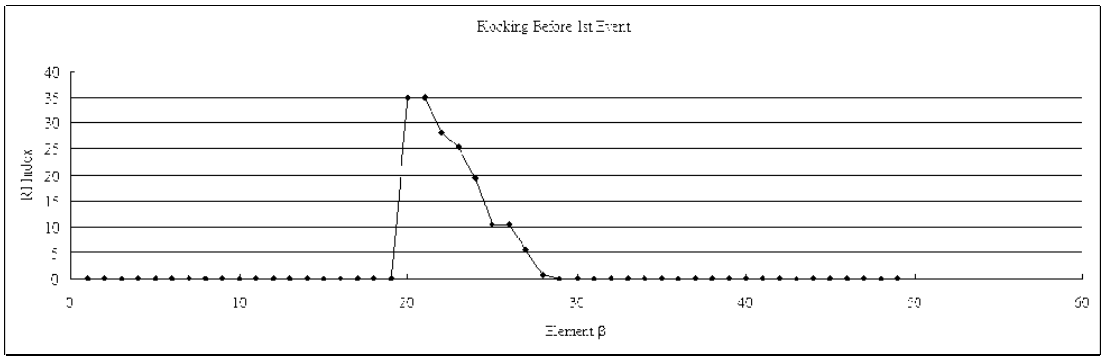


Figure 5-16. Blocking between Inlet and 1st Event – RI Chart

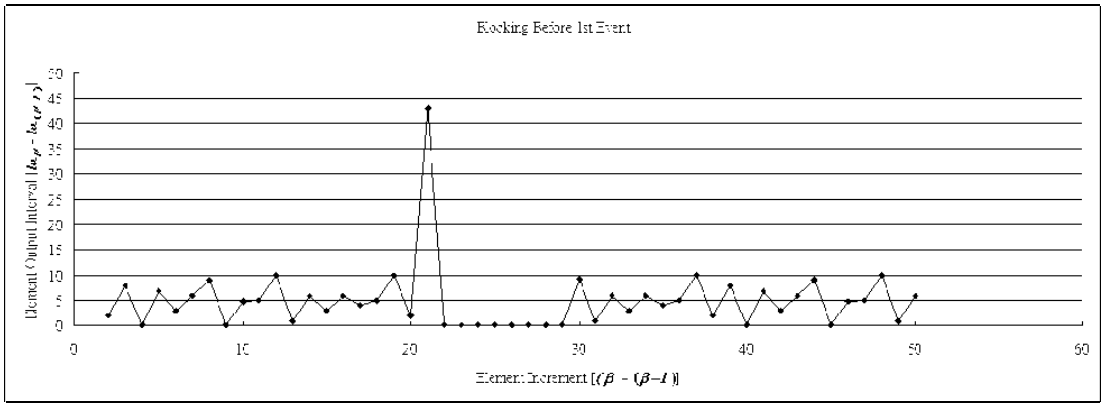


Figure 5-17. Blocking between Inlet and 1st Event – ILT Chart

Figure 5–16, **5–18**, **5–20** and **5–22** are the RI charts at the four locations defined before. The time span of this ramp is the effective time of the Blocking. The initiation of the Blockings was the same for all of them, but we can see that the effect on the ROI came later when the location of a Blocking was closer to the inlet.

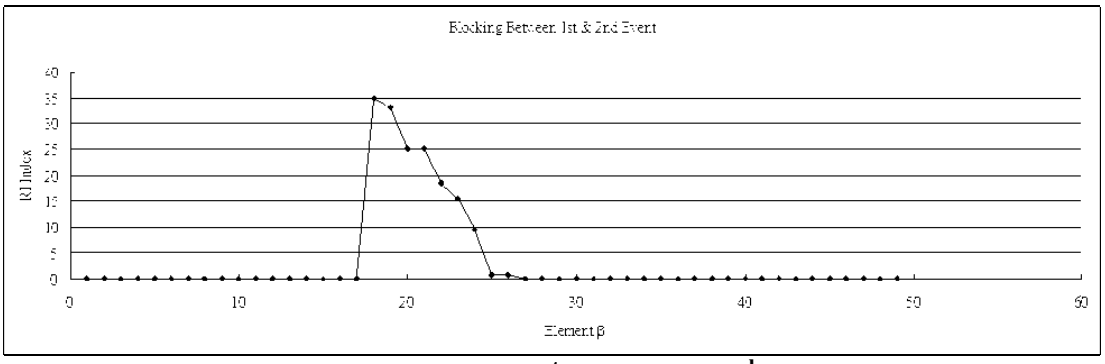


Figure 5-18. Blocking between 1st Event and 2nd Event – RI Chart

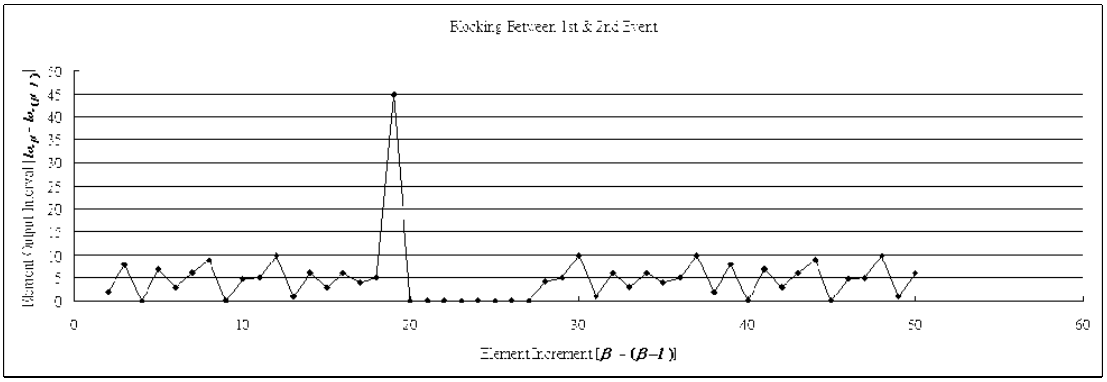


Figure 5-19. Blocking between 1st Event and 2nd Event – ILT Chart

In terms of ILT charts, **Figures 5–17**, **5–19**, **5–21** and **5–23** are the four which correspond to the experiment. These charts give information regarding time while a RI chart has to wait for an element to turn up before any data can be plotted on that chart and this may take a very long time. However, the ILT chart can alert staff when an error occurs, when there is a difference between the plotted pattern and the master signature. If the repeated pattern is a horizontal line with a value equal to (T_g), then the system has still not yet recovered. The system has fully recovered once a repeated pattern appears again. An example has been shown in Figure 5-21, it shows an ILT chart showing the scenario of blocking, recovery, and full recovery.

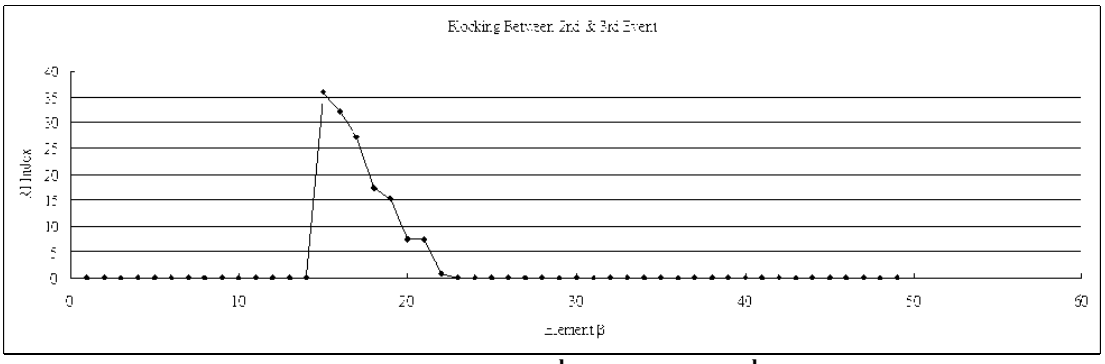


Figure 5-20. Blocking between 2nd Event and 3rd Event – RI Chart

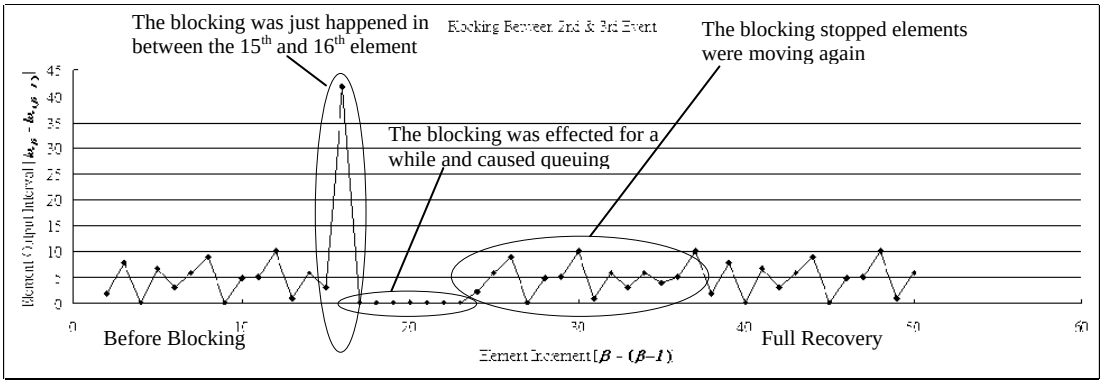


Figure 5-21. Blocking between 2nd Event and 3rd Event – ILT Chart

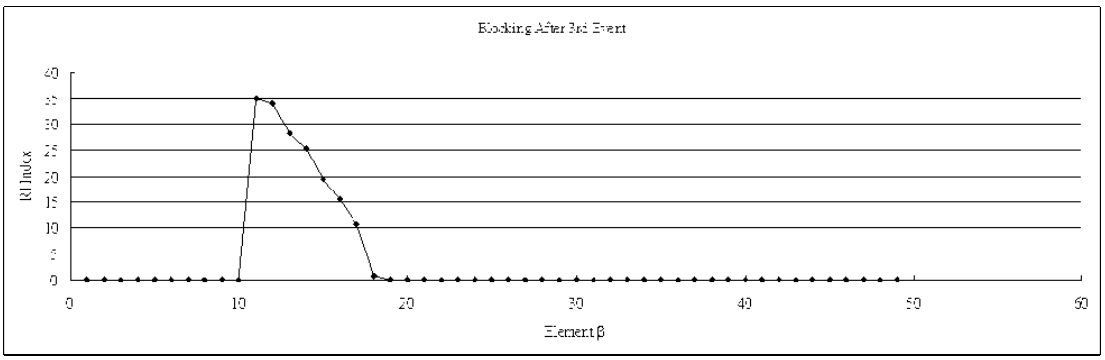


Figure 5-22. Blocking between 3rd Event and Outlet – RI Chart

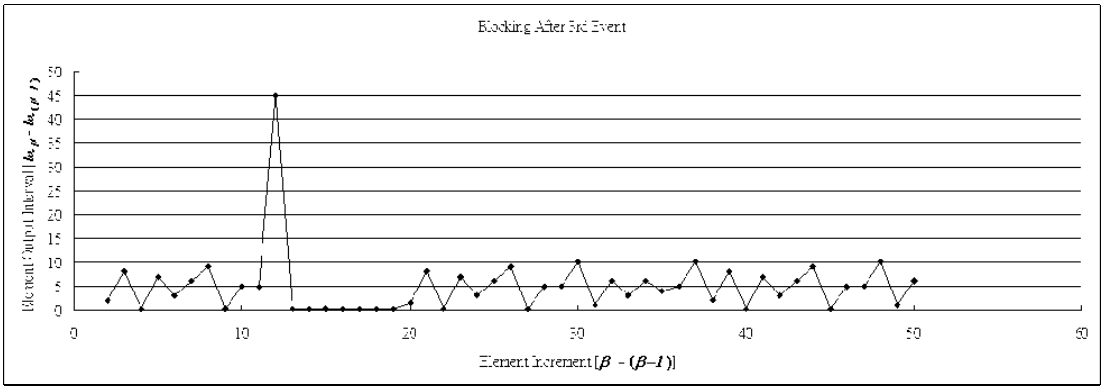


Figure 5-23. Blocking between 3rd Event and Outlet – ILT Chart

5.3.3 Unknown Event Occurrence in ROI

In this test, the four positions are the same as in the previous test. The disturbance in this case is the Unknown Event Occurrence, which is self-explanatory. This can be simulated by introducing an additional event but it is hidden from the monitoring core; the settings are given in **Table C–17** and the test data are listed in **Tables C–18** to **C–21** in **Appendix-C**.

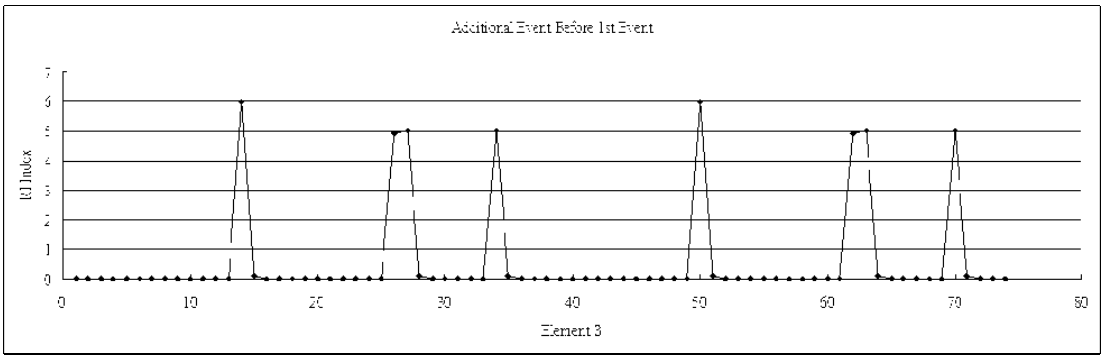


Figure 5-24. Unknown Event between Inlet and 1st Event – RI Chart

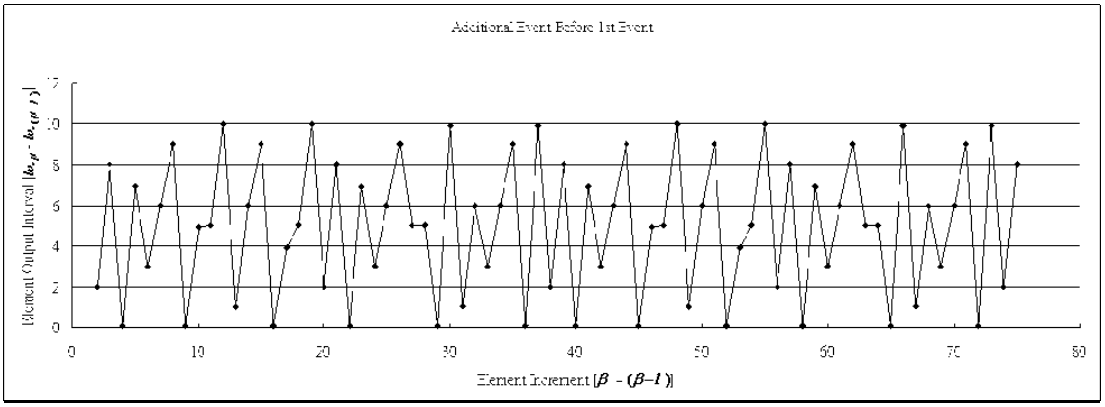


Figure 5-25. Unknown Event between Inlet and 1st Event – ILT Chart

The associated RI charts are presented in **Figures 5–24**, **5–26**, **5–28** and **5–30**. Here, the repeated cycle on the RI chart may not last for the same duration as the master signature (T_x); sometimes, it can be the same, if the cycle duration of the unknown event happens to fall into a factor of that particle ROI LCM calculated.

In the test case, the new LCM values are not the same and it consists of 180 time units instead of 90 time units. Therefore, the elements involved in a cycle are 36 elements/cycle.

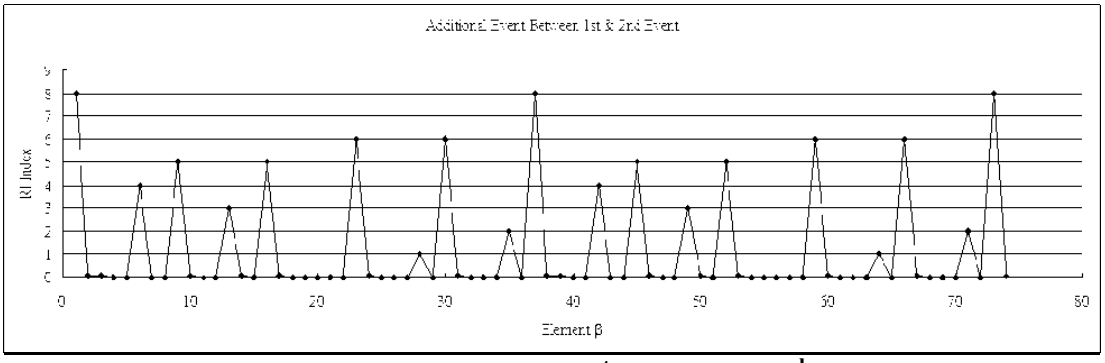


Figure 5-26. Unknown Event between 1st Event and 2nd Event – RI Chart

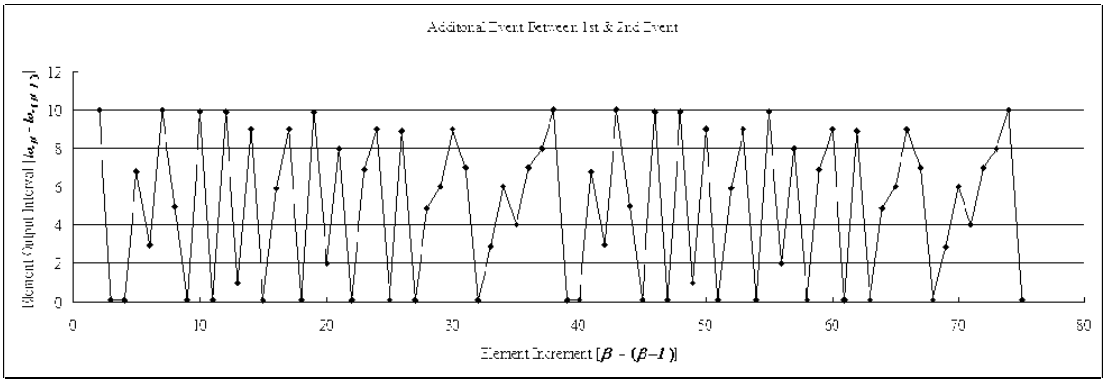


Figure 5-27. Unknown Event between 1st Event and 2nd Event – ILT Chart

The lowest value on the RI chart will be zero but the Slowdown case may not be. The repeated cycle durations shown on the ILT charts are the same as those on the RI charts. Moreover, the number of points that hit the (T_g) value are more than that of the Normal setting. The related ILT charts are presented in **Figures 5–25**, **5–27**, **5–29** and **5–31**.

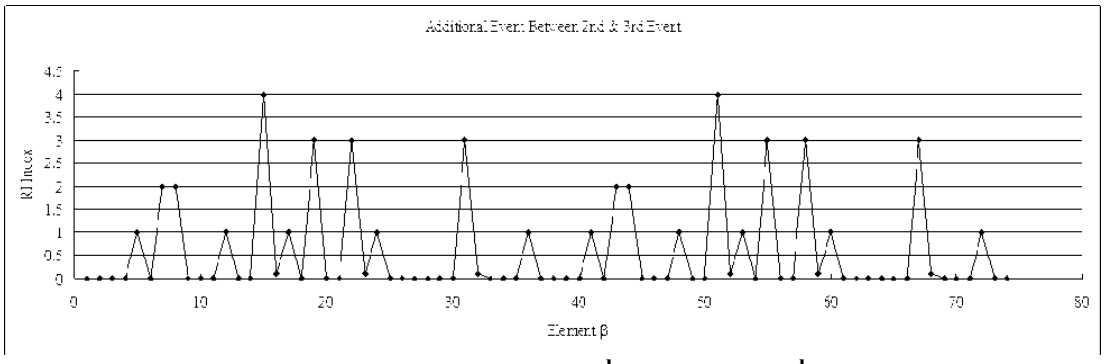


Figure 5-28. Unknown Event between 2nd Event and 3rd Event – RI Chart

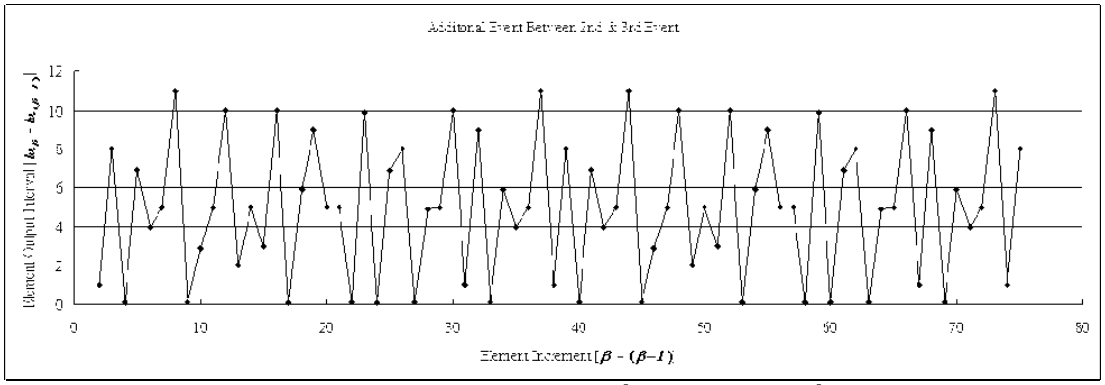


Figure 5-29. Unknown Event between the 2nd Event and 3rd Event – ILT Chart

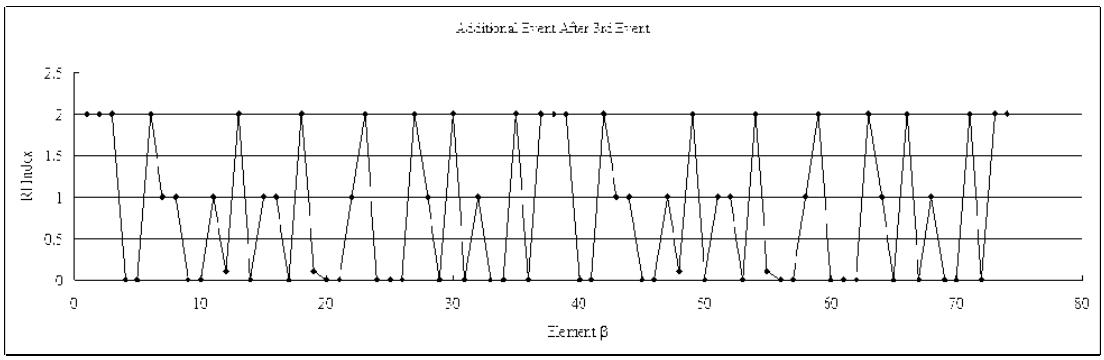


Figure 5-30. Unknown Event between Outlet and 3rd Event – RI Chart

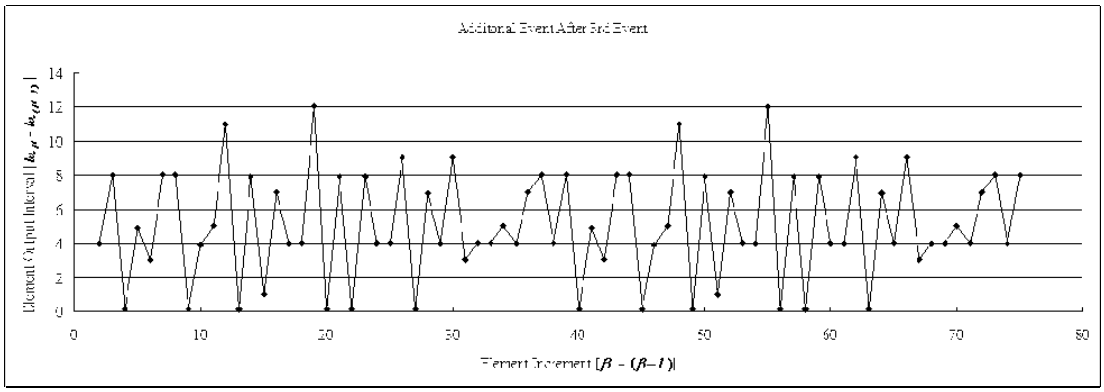
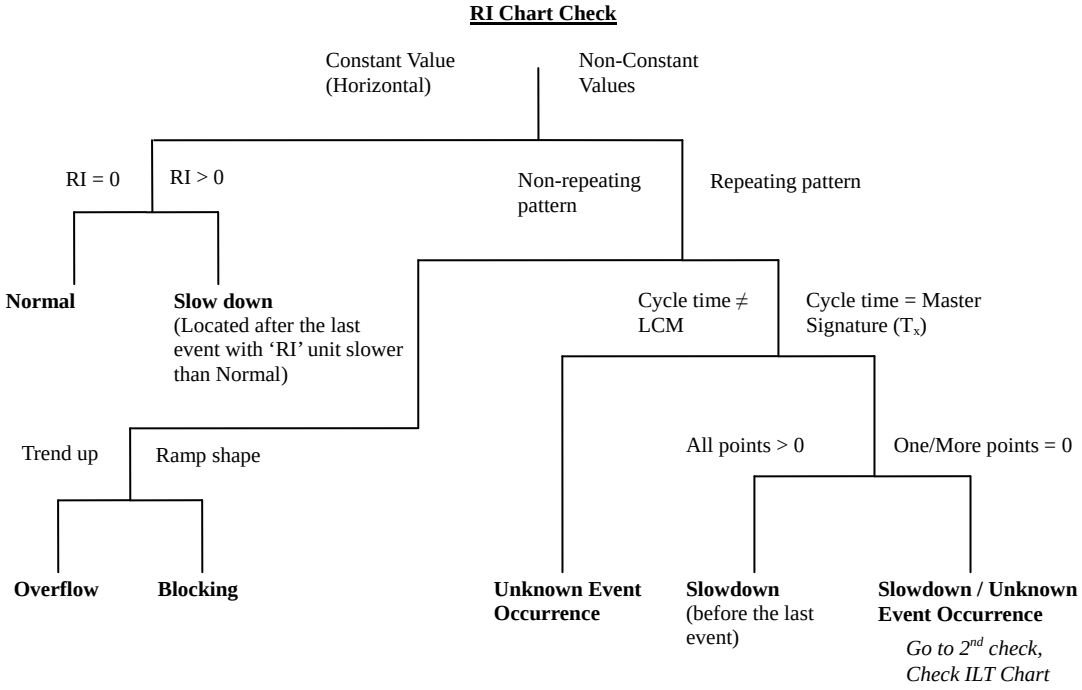


Figure 5-31. Unknown Event between Outlet and 3rd Event – ILT Chart

5.4 Reasoning Approach

Based on the results of the previous studies, it is observed that the RI and ILT charts can be employed to monitor the conditions in a ROI. In summary, the reasoning approach is that one should check the RI chart first and then the ILT chart, if it is necessary. The inspection tactic is outlined in the **Figures 5–32** and **5–33**. **Figure 5–32** outlines the logic to be applied in the first check. Most of the abnormal conditions can be identified but sometimes the Unknown Event Occurrence and the Slowdown may not be easily distinguished. If this turns out to be the case, it is necessary to go through the second check on the coupled ILT chart with reference to **Figure 5–33**. A corresponding rundown table describing the conditions against the observations on the two charts has also been provided in **Table 5–3**.



**The system can be expressed as tending to errors if the inter-element leaving time of the testing is longer than the max. amplitude of the master signature (ILT Chart).*

Figure 5-32. The First Check (RI Chart)

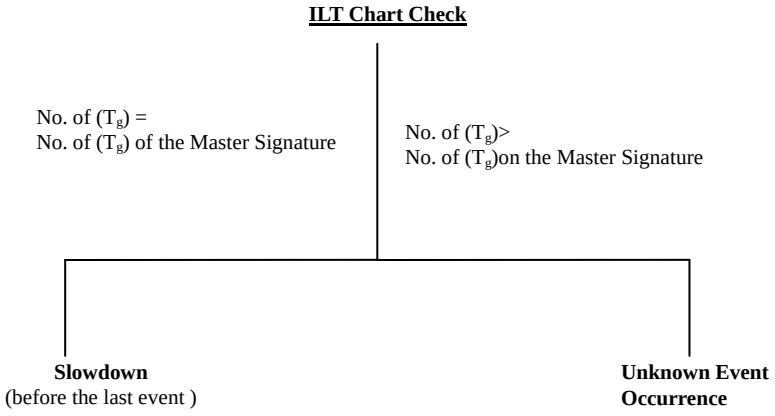


Figure 5-33. The Second Check (ILT Chart)

Table 5-3. ROI Diagnosis Table

	RI & ILT Pattern	Possible Error	Possible Location
1	Only '0s' on RI Chart	Running normally	---
2	Inter-element leaving time of the measuring pattern > max. amplitude of ILT master signature.	Tend to abnormal	---
3	A trend up pattern on RI Chart	Overflow	---
4	Constant value on RI Chart	Slowdown	Slowdown after last event
	Repeated pattern on RI Chart. The cycle period = master signature and the minimum point > 0		Slowdown before last event
	Repeated pattern on ILT Chart = Master Signature Or No. of T_g of repeated pattern on ILT Chart = No. of T_g of the master signature		
5	One-off ramp pattern on RI Chart	Blocking	---
6	Repeated pattern on RI Chart. The cycle period ≠ Master Signature	Unknown Event Occurrence	---
	The no. of T_g of the repeating pattern > The no. of T_g of the master signature		

6 Chapter Six – Discussion and Further Work

The reasoning function is used to demonstrate the use of the mathematical model developed in **Chapter 3** of this research. The proposed model was demonstrated to be useful in regular system, for example a production system (Regular input rate, constant (T_e)). On the other hand, it has potential for being used in other kinds of systems as well. In this section, first, what we have learned in this research is discussed, next the possible applications of the proposed model are examined, and finally we look at the proposed directions for further research work.

6.1 Other Supplementary benefits of Reasoning

The condition analysis method has been discussed and the results of using it are shown in Chapter 5. The various types of errors in the ROI can be identified as long as certain patterns can be recognized by the first and the second checks from the RI and the ILT charts. However, some of the uncertain information such as the location and the degree of impact of the error may also be inferred, for instance, the Slowdown error which occurs after the last event. Even though the exact location where the error occurred can not be detected clearly, it can be deduced by comparing the known locations. For example, if the same Blocking error occurred at two different ROI locations at the same time, the time when the errors affected the ROI would be different. The result shows that the error which occurred at a location closest to the output of the ROI, makes its influence felt earlier than the error which occurred at a location further from the output of the ROI. Such phenomenon can be seen from the **Figures 5–16 to 5–23**. Besides, the degree of impact may be determined by

measuring the length of the blocking time. According to the same figures (**Figures 5–16 to 5–23**), the estimated blocking period is somewhere between 35 and 45 time units. When comparing the value of the estimated blocking period to the blocking error setup in **Table C–12** at **Appendix-C**, the blocking setup is in 40 time units. This shows that the estimation is accurate. Similar information is also applicable to the Slowdown error as well. For example, the average RI values (within one cycle) of **Figures 5–8**, **5–10** and **5–12** are in 2.89 time units. According to the Slowdown error setup (Phase Compensation factor) is in 3 time units, as can be seen from **Table C–7** in **Appendix-C**. The results also show that the estimation is accurate.

6.2 Irregular Element Arrival

Analysis of regular input conditions has been discussed in Chapter 5. This section discusses how the reasoning effect can be useful when a system does not have any regular input. The reasoning method is generally useful for regular systems. It needs, a regular element input time period (constant (T_e)). When the element input time is no longer constant but the ROI is still running normally, the pattern that is shown on the RI chart should remain as a horizontal line across RI = 0. For the non-constant element input time cases, part of the first check rule (see **Figure 5–32**) is still valid for checking the conditions of the ROI. However, when a repeated cyclic pattern appears, the second check may not be so useful as it is difficult to identify repeating patterns., The calculation of a repeated cycle (T_x) is not valid the second time round because (T_e) is not a constant value, see **Equation 12** in **Chapter 3**. **Figures 6–1** to **6–3** are the ILT chart examples to show if the input rate of a system bears a certain degree of tolerance; 1%, 5% and 10% (see **Tables C–22** to **C–24** at **Appendix-C** for the random

simulation data). The dotted lines on the figures are the master signature of the system, and the solid lines are the responses if the (T_e) is bearing tolerance. The data for constructing **Figures 6–1** to **6–3** is listed in **Tables C–25** to **C–27** at **Appendix-C**. It can be seen that, the solid line deviates more seriously from the dotted line when the (T_e) bears a larger tolerance. The chart with lower tolerance, for example **Figures 6–1** (with 1% tolerance) is still able to identify the repeating cycle. On the other hand, the others can not. If the input rate is irregular, the repeating cycle cannot be identified. The possibility of making improvements in such cases will be discussed in **Section 6.4** which deals with suggestions for further research work.

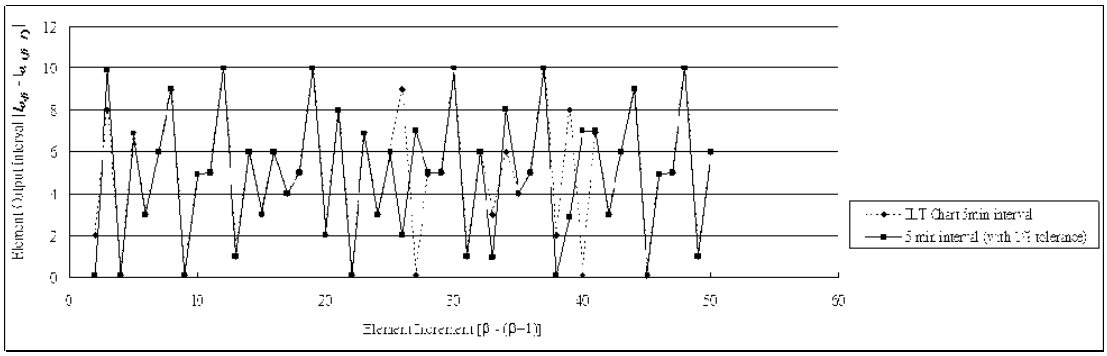


Figure 6-1. Example of ILT Chart – T_e with 1% Tolerance

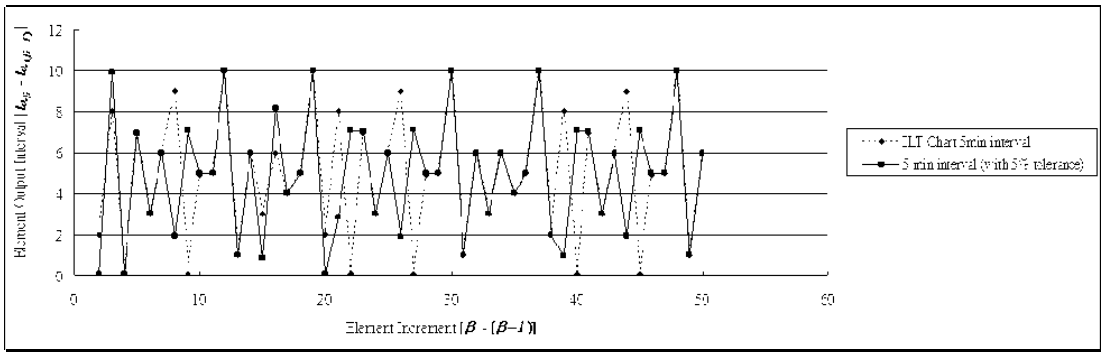


Figure 6-2. Example of ILT Chart – T_e with 5% Tolerance

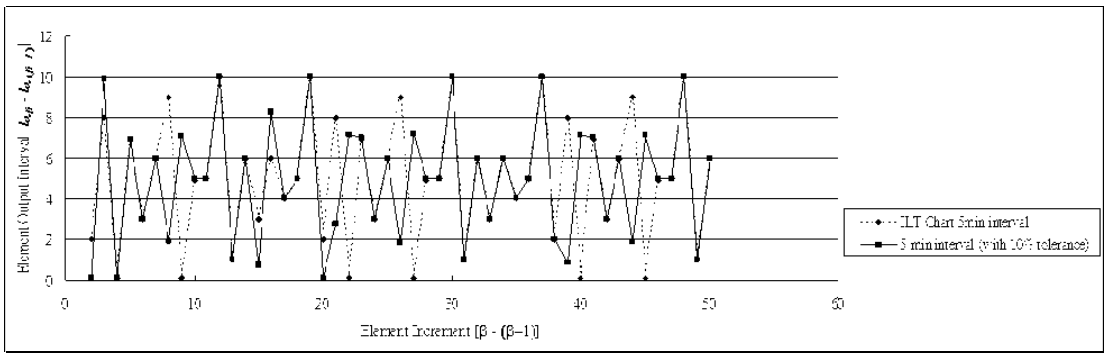


Figure 6-3. Example of ILT Chart – T_e with 10% Tolerance

6.3 Possible Applications

This section demonstrates two possible applications of the proposed model. The first one is used to solve a dynamic shortest path problem with delays on each node; the next example is a bus catching example which helps passengers to be at the bus station in time to catch the bus.

The proposed model can be used to solve a dynamic shortest path problem by determining the element output time. The Graph Theory was introduced by Leonhard Euler in 1736 (West, 2001) and it can also be engaged in solving the Shortest Path Problem (SSP). The prime aim of SPP is to find a path between two nodes such that the sum of the weights of its constituent nodes is minimized (Ahuja *et al.*, 1993). An example of the shortest path problem is finding the quickest way to get from one location to another on a road map.

Figure 6–4 shows an example of a SPP which contains 9 routes. Those routes are unidirectional in terms of traveling. They start from point A and go to the destination F. The routes can be the railway train routes or any other transportation routes. The

average travel distances (T_s), in terms of time, are listed in the second column of **Table 6–1**. This table also indicates traditional SPP and considers the route in a static manner, in that the route that contains the least traveling distance is determined as the shortest path. Based on such analysis theorem, the shortest path is A-B-D-F. It takes the least traveling time 14:30:00 (hr.:min.:sec.). Does the traveling condition vary against time? Should we also consider the number of runs of the scheduled train as well? The proposed method is able to handle such dynamic conditions. It is able to take the number of runs of the scheduled train into consideration if necessary. Indeed, this example was taking the departing frequency in every 10 mins and therefore, cannot be considered as an exhaustive search.

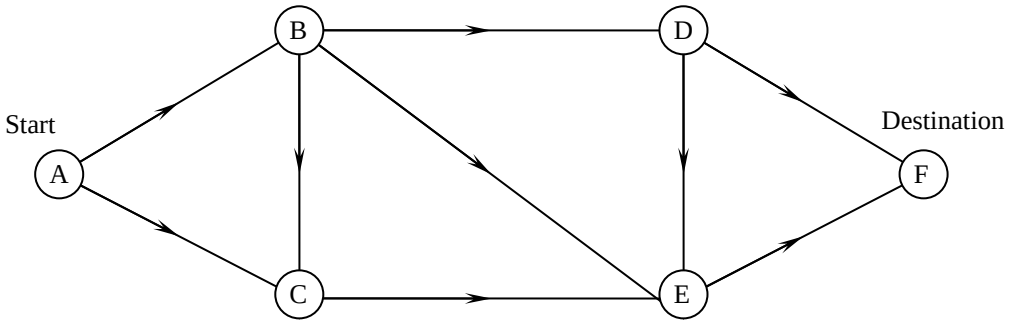


Figure 6-4. Traveling Routes for a shortest path problem

The columns under the headings Event 1 and Event 2 of the **Table 6–1** indicates the parameters of the dynamic behaviors – events that happen en route. The 1st Events of all nodes are located at the very beginning of each route and which can be represented as the number of runs of the scheduled vehicle, i.e. the regular departure cycle of the train schedules (departure every 1.5hrs, etc.). The 2nd Event, can be represented as the traffic lights. It is located somewhere along every route and the locations are determined by the (T_k). Please refer to Chapter 3 for the details of each parameter. In this example, (T_g) can be neglected since passengers are travelling on the same train, so there is no queuing effect.

Table 6-1. Route Information

			1st Event				2nd Event			
	Routes	T_s (hr.:min.:sec.)	T_1 (hr.:min.:sec.)	$T_{pc,1}$ (hr.:min.:sec.)	$T_{po,1}$ (hr.:min.:sec.)	$t_{p,1}$ [10-Oct-2008] (hr.:min.:sec.)	T_2 (hr.:min.:sec.)	$T_{pc,2}$ (hr.:min.:sec.)	$T_{po,2}$ (hr.:min.:sec.)	$t_{p,2}$ [10-Oct-2008] (hr.:min.:sec.)
1	A-B	04:00:00	00:00:00	00:50:00	00:00:01	07:00:00	02:00:00	00:40:00	00:20:00	08:00:00
2	A-C	05:00:00	00:00:00	00:30:00	00:00:01	07:00:00	02:00:00	00:10:00	00:30:00	07:00:00
3	B-C	05:30:00	00:00:00	00:40:00	00:00:01	07:30:00	02:00:00	00:20:00	00:00:00	07:30:00
4	B-D	04:30:00	00:00:00	00:60:00	00:00:01	07:30:00	02:00:00	00:20:00	00:60:00	07:30:00
5	B-E	06:30:00	00:00:00	00:10:00	00:00:01	06:30:00	02:00:00	00:20:00	00:00:00	06:30:00
6	C-E	06:00:00	00:00:00	00:20:00	00:00:01	08:00:00	02:00:00	00:40:00	00:80:00	08:00:00
7	D-E	06:30:00	00:00:00	00:40:00	00:00:01	07:00:00	02: 0:00	00:30:00	00:10:00	07:00:00
8	D-F	06:00:00	00:00:00	00:50:00	00:00:01	07:30:00	02:00:00	00:20:0	00:10:00	07:30:00
9	E-F	04:30:00	00:00:00	00:20:00	00:00:01	08:00:00	02:00:00	00:10:00	00:10:00	08:00:00

Static shortest path: A-B-D-F, duration *14:30:00.* (hr.:min.:sec.)

The purpose of this example is to demonstrate the ability of the model. Currently it is not easy to find an effective algorithm for handling such dynamic SPP with such events. Nevertheless, we can still calculate and analyze the optimal routes (against time) using the proposed model. This method first enumerates all the possible combinations of possible paths, then sorts out the path with nearest time to the next destination. Hence, each combination can be solved by the algorithm discussed in **Chapter 3**. Finally, the dynamic shortest path is suggested as a result.

Table 6–2 is the analyzed summary on the date 10-Oct-2008 of the train transport service from 06:40:00 until 18:20:00. It is supposed that passengers will arrive at the node A every 10 mins. The results show that once the dynamic factors have been taken into consideration, the duration of the shortest path can vary from +2.3% to 9.2% if only the number of runs only are considered (1^{st} event only). The difference can be maximized up to 13.8% if both the number of runs and the presence of traffic lights (1^{st} plus 2^{nd} Events) are taken into consideration. Furthermore, the result also shows that the dynamic shortest path varies against time instead of being a fixed route. A passenger can decide the travel route in such a timely way so as to take the fastest travel route. This is the main difference between the dynamic and the static way of solving the shortest path problem.

Table 6-2. The Results of the Train Dynamic Route Analysis

		Presence of Event 1			Presence of Events 1&2		
	Start time / $t_{i,\beta}$ [10-Oct-2008] (hr.:min.:sec.)	Duration (hr.:min.:sec.)	Leaving time / $t_{o,\beta}$ [10-Oct-2008] (hr.:min.:sec.)	Shortest path	Duration (hr.:min.:sec.)	Leaving time / $t_{o,\beta}$ [10-Oct-2008] (hr.:min.:sec.)	Shortest path
1	06:40:00	15:20:00	22:00:00	ABEF	16:00:00	22:40:00	ABDF
2	06:50:00	15:10:00	22:00:00	ABEF	15:40:00	22:40:00	ACEF
3	07:00:00	15:30:00	22:30:00	ACEF	15:50:00	22:30:00	ABEF
4	07:10:00	15:40:00	22:50:00	ABEF	16:20:00	22:50:00	ACEF
5	07:20:00	15:30:00	22:50:00	ABEF	16:10:00	23:30:00	ABDF
6	07:30:00	15:20:00	22:50:00	ABEF	16:00:00	23:30:00	ABDF
7	07:40:00	15:20:00	23:00:00	ABEF	15:50:00	23:30:00	ABDF
8	07:50:00	15:10:00	23:00:00	ABEF	15:10:00	23:30:00	ABEF
9	08:00:00	15:30:00	23:30:00	ACEF	15:40:00	23:00:00	ABEF
10	08:10:00	15:40:00	23:50:00	ABEF	15:50:00	23:40:00	ABEF
						[11-Oct-2008]	
11	08:20:00	15:30:00	23:50:00	ABEF	15:40:00	0:00:00	ABEF
12	08:30:00	15:20:00	23:50:00	ABEF	15:30:00	0:00:00	ABEF
13	08:40:00	15:00:00	23:40:00	ABEF	15:00:00	0:00:00	ABEF
			[11-Oct-2008]				
14	08:50:00	15:30:00	0:20:00	ABDF	15:30:00	0:20:00	ABDF
15	09:00:00	15:20:00	0:20:00	ABDF	15:20:00	0:20:00	ABDF
16	09:10:00	15:10:00	0:20:00	ABDF	15:10:00	0:20:00	ABDF
17	09:20:00	15:00:00	0:20:00	ABDF	15:00:00	0:20:00	ABDF
18	09:30:00	14:50:00	0:20:00	ABDF	14:50:00	0:20:00	ABDF
19	09:40:00	15:30:00	1:10:00	ABDF	15:30:00	1:10:00	ABDF
20	09:50:00	15:20:00	1:10:00	ABDF	15:20:00	1:10:00	ABDF
21	10:00:00	15:10:00	1:10:00	ABDF	15:10:00	1:10:00	ABDF
22	10:10:00	15:00:00	1:10:00	ABDF	15:00:00	1:10:00	ABDF
23	10:20:00	14:50:00	1:10:00	ABDF	14:50:00	1:10:00	ABDF
24	10:30:00	15:30:00	2:00:00	ABDF	15:30:00	2:00:00	ABDF
25	10:40:00	15:20:00	2:00:00	ABDF	15:20:00	2:00:00	ABDF
26	10:50:00	15:10:00	2:00:00	ABDF	15:10:00	2:00:00	ABDF
27	11:00:00	15:00:00	2:00:00	ABDF	15:00:00	2:00:00	ABDF
28	11:10:00	14:50:00	2:00:00	ABDF	14:50:00	2:00:00	ABDF
29	11:20:00	15:50:00	3:10:00	ABEF	16:20:00	3:40:00	ABEF
30	11:30:00	15:40:00	3:10:00	ABEF	15:40:00	3:10:00	ACEF
31	11:40:00	15:20:00	3:00:00	ABEF	15:40:00	3:20:00	ABEF
32	11:50:00	15:10:00	3:00:00	ABEF	15:40:00	3:30:00	ABEF
33	12:00:00	15:00:00	3:00:00	ABEF	15:00:00	3:00:00	ABEF
34	12:10:00	15:40:00	3:50:00	ABEF	16:30:00	4:40:00	ACEF
35	12:20:00	15:30:00	3:50:00	ABEF	16:00:00	4:20:00	ACEF
36	12:30:00	15:20:00	3:50:00	ABEF	15:50:00	4:20:00	ACEF
37	12:40:00	15:20:00	4:00:00	ABEF	15:50:00	4:30:00	ABDF
38	12:50:00	15:10:00	4:00:00	ABEF	15:40:00	4:30:00	ABDF
39	13:00:00	15:30:00	4:30:00	ACEF	15:30:00	4:30:00	ACEF
40	13:10:00	15:40:00	4:50:00	ABEF	15:50:00	5:00:00	ABEF
41	13:20:00	15:30:00	4:50:00	ABEF	16:00:00	5:20:00	ABDF
42	13:30:00	15:20:00	4:50:00	ABEF	15:30:00	5:00:00	ABEF
43	13:40:00	15:00:00	4:40:00	ABEF	15:20:00	5:00:00	ABEF
44	13:50:00	15:30:00	5:20:00	ABDF	15:30:00	5:20:00	ABDF
45	14:00:00	15:20:00	5:20:00	ABDF	15:20:00	5:20:00	ABDF
46	14:10:00	15:10:00	5:20:00	ABDF	15:10:00	5:20:00	ABDF
47	14:20:00	15:00:00	5:20:00	ABDF	15:00:00	5:20:00	ABDF
48	14:30:00	14:50:00	5:20:00	ABDF	14:50:00	5:20:00	ABDF
49	14:40:00	15:30:00	6:10:00	ABDF	15:30:00	6:10:00	ABDF
50	14:50:00	15:20:00	6:10:00	ABDF	15:20:00	6:10:00	ABDF
51	15:00:00	15:10:00	6:10:00	ABDF	15:10:00	6:10:00	ABDF
52	15:10:00	15:00:00	6:10:00	ABDF	15:00:00	6:10:00	ABDF
53	15:20:00	14:50:00	6:10:00	ABDF	14:50:00	6:10:00	ABDF
54	15:30:00	15:30:00	7:00:00	ABDF	15:30:00	7:00:00	ABDF
55	15:40:00	15:20:00	7:00:00	ABDF	15:20:00	7:00:00	ABDF
56	15:50:00	15:10:00	7:00:00	ABDF	15:10:00	7:00:00	ABDF
57	16:00:00	15:00:00	7:00:00	ABDF	15:00:00	7:00:00	ABDF
58	16:10:00	14:50:00	7:00:00	ABDF	14:50:00	7:00:00	ABDF
59	16:20:00	15:50:00	8:10:00	ABEF	16:00:00	8:20:00	ACEF
60	16:30:00	15:40:00	8:10:00	ABEF	16:10:00	8:40:00	ABDF
61	16:40:00	15:20:00	8:00:00	ABEF	16:00:00	8:40:00	ABDF
62	16:50:00	15:10:00	8:00:00	ABEF	15:50:00	8:40:00	ABDF
63	17:00:00	15:00:00	8:00:00	ABEF	15:30:00	8:30:00	ACEF
64	17:10:00	15:40:00	8:50:00	ABEF	16:20:00	9:30:00	ACEF
65	17:20:00	15:30:00	8:50:00	ABEF	16:10:00	9:30:00	ABDF
66	17:30:00	15:20:00	8:50:00	ABEF	16:00:00	9:30:00	ABDF
67	17:40:00	15:20:00	9:00:00	ABEF	15:50:00	9:30:00	ABDF
68	17:50:00	15:10:00	9:00:00	ABEF	15:10:00	9:00:00	ABEF
69	18:00:00	15:30:00	9:30:00	ACEF	15:40:00	9:40:00	ABEF
70	18:10:00	15:40:00	9:50:00	ABEF	15:50:00	10:00:00	ABEF
71	18:20:00	15:30:00	9:50:00	ABEF	15:40:00	10:00:00	ABEF
	Min.:	*14:50:00*	*(+2.3%)*	*Min.:*	*14:50:00*	*(+2.3%)*	
	Max.:	*15:50:00*	*(+9.2%)*	*Max.:*	*16:30:00*	*(+13.8%)*	

The second example is about arriving at the bus stop in time to catch the bus. We will call it, on-time bus catching. The purpose of this example is to demonstrate the function of the discrete time distributions. It is supposed that the bus travelling system has not installed any measuring devices at the events (i.e. traffic lights) and therefore, the estimated cycle initiation times are possibly in discrete distributions. So, it is necessary to consider the initiation time of the events (see **Chapter 3, Section 3.5** for details). **Figure 6–5** illustrates a Bus route ROI where a bus departs from a terminal at the left side and travels to a bus stop at the right hand side. Three events, three sets of traffic lights, are located at three different places between the bus terminal and the bus stop.

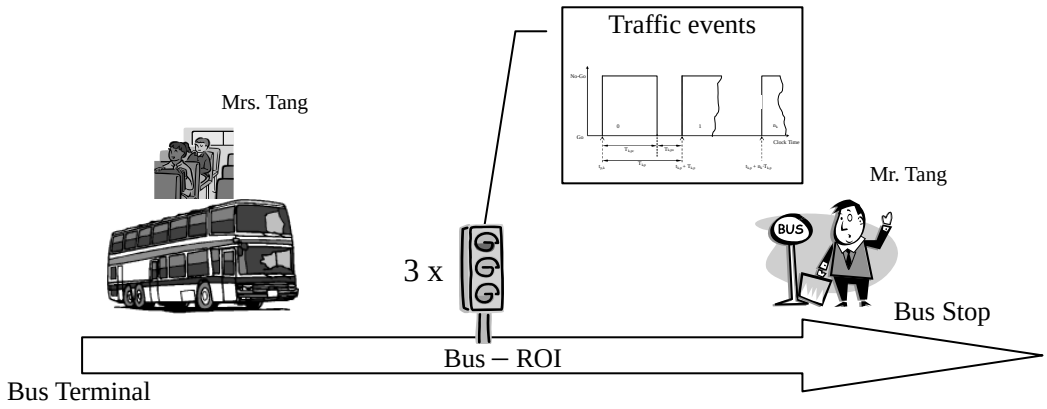


Figure 6-5. Bus Catching Example

Mr. Tang is running to the bus stop and expects to catch the bus there at the time it is scheduled to arrive which is at 18:24:00. Mrs. Tang is already on bus which departed from the terminal at 18:05:00. Can Mr. Tang meet Mrs. Tang on the bus? We may use the proposed model to conduct an analysis, see below:

- Bus schedule (departure from bus terminal): 18:05:00; 18:30:00; …;
- The net travelling time (T_s) from bus terminal to bus stop: 15 min. (net, traffic free);

- The locations of the 3 sets of traffic lights are 2 min. apart from each other, the first one is 2 min. away from the bus terminal;
- Assume that the Bus – ROI has no other traffic. The bus is alone on the road.

The section of road along which the bus travels from the bus terminal to bus stop can be defined as Bus – ROI. There are three sets of traffic lights along this section of the road and it is assumed that there is no traffic ahead of the bus. These three traffic lights can be represented by three events. The ROI parameters are listed in **Table 6–3**. The program analysis result is listed in **Figure D–4** (Colum 'S') at **Appendix D**. A table to illustrate the possible time that Mr. Tang can meet his wife is listed in **Table 6–4**. The result shows that he has a 40% chance (21.25+0.75+18) of failing to meet his wife. If he wants to be 100% certain of meeting her on bus, he must hurry up and must arrive at the bus stop no later than 18:22:45.

Table 6-3. ROI Configuration – Bus Catching Example

ROI Configuration	
T_s (min.)	15
T_g (min.)	0.25
Event Cycle Definition	

k	$T_{k,pc}$ (min.)	$T_{k,po}$ (min.)	T_k (min.)	$t_{k,p,m}$ (min.)	$P(t_{k,p,i})$
1	1.5	2	2	0	0.95
				-0.3	0.05
2	2	0.5	4	0	0.75
				-0.5	0.25
3	1	1	6	0	0.8
				-1	0.15
				0.5	0.05

Table 6-4. Predicted possible bus arrival time

Possibility (%)	O'clock (pm)
21.25	18:22:45
0.75	18:23:00
18	18:23:15
3	18:24:00
57	18:24:15

The proposed model is not limited to being applied to the above examples, it can also be applied to other cases which can be represented by a ROI configuration and which contain physical element flows, usually flowing in a regular manner. A production line is one such regular system. For example, the balance of productivity against system throughput time, is a typical problem of designing a production line. A similar analysis can be used to shorten the throughput time in order to maximize the usage of a production line. However, it may reduce the productivity. The balance setting, and adjusting the input arrival (T_e), can be simulated by using the proposed model.

6.4 Further Work

This research initiates a new way of thinking about how to determine the conditions of a dynamics system by measuring the input and output time of the system. The concept can be modified by sophisticated mathematical models so that more complicated applications may be handled. This section proposes a few possible enhancements, which perhaps can be the objectives of further research.

Firstly, the parameters of the event cycles may be rearranged in a periodical manner, so that the ($T_{k,p}$), ($T_{k,pc}$) and ($T_{k,po}$) are no longer constant values. They will correspond to time. The periodic characteristics of those parameters could be predefined or obtained by measurements. Secondly, the initial time of the cycles ($t_{p,k}$)

can be a continuous distribution instead of a discrete distribution. Thirdly, the boundaries of the effective and non-effective time of events can contain certain tolerances in terms of continuous distributions, see **Figure 6–6**. If the above proposals were implemented, many probability scenarios would occur. For such cases, simple mathematical calculations and simple programming would no longer be an adequate approach, more intelligent computational methods may be required not only for determining the output time (or time series), but also for analyzing the conditions. Recently, many intelligent methods for recognizing patterns have been developed by scholars and researchers. Some of them have been reviewed in **Chapter 2**. Perhaps this can be another research for scholars who are interested in further developing it. The concept of this research project may finally be constructed as a software package for use in logistics.

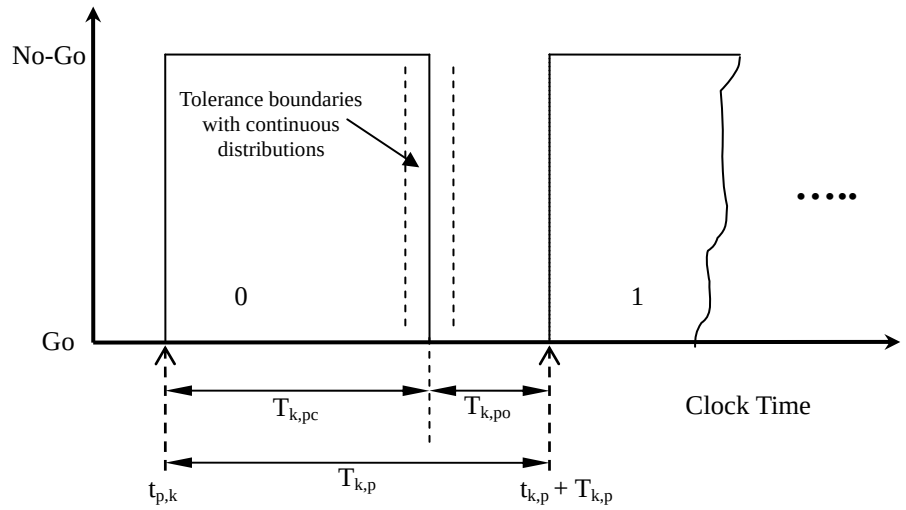


Figure 6-6. Concepts of Event Cycles

6.5 Other Possible Approaches

The proposed reasoning modeling is a time based method which is based on checking the deviations between the deterministic arrivals and the measured time at a check point, so that the condition of a ROI and the entire system can be analyzed. However, it may not be the only possible approach. Perhaps the Queuing theory and the Fluid Mechanical study approaches might also be other possible ways to solve such kind of problems. The details of these alternative approaches have not been studied in this thesis, but some alternative concepts have been mentioned in this section. Readers may dig into them more deeply if they have the interest.

In the Queuing theory approach, one's system is generally assumed to be in a state of equilibrium. On the other hand, the problems discussed in this research operate in a dynamic manner and the system may never reach equilibrium. So the traditional Queuing theory may not be so useful to handle such kind of cases. If some other new algorithm is able to manipulate the dynamic situation, another research project may also be initiated. Besides the Queuing theory, the element flow can be assumed as a fluid dynamic flow pipe with a buffer of a certain size, see **Figure 6–7**. It simulates a process which has input from the left flowing in at certain rate. The center is a buffer and the right is the exit with a certain output rate. For this approach, the definition of pressure, and the other related functions need to be redefined by sets of experiments.

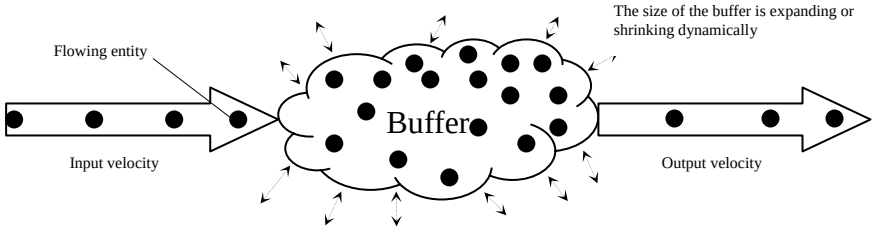


Figure 6-7. Conceptual System Element Flow Line

7 Chapter Seven - Conclusion

In this research, the first objective is to form a modular framework for monitoring dynamic systems. This has been done by applying the concept of identifying the stream or flow of representative elements in the system and this led to the establishment of Regions of Interests (ROI). Then, with the introduction of a pair of simple counting devices for each ROI, one at the inlet and another at the outlet, the conditions in the ROI were examined.

The second objective is to formulate algorithms to transform the incoming data into useful information. Subsequently, mathematical algorithms have been generated for each ROI and two charts, RI chart and ILT chart have been introduced to extract the embedded information. It is observed that the ILT chart produces repeated patterns and under normal circumstances, the pattern generated is called the master signature. This master signature is a very important representation of the condition of the ROI and it should be obtained at the very beginning stage. However, it is also noted that a ROI that producing cyclic pattern identical to master signature may not mean it is healthy at all as a slowdown just before the outlet gives a pattern like the master signature but this can be detected in the RI chart. Additionally, in case the Event initiation time is a set of discrete probability, a learning period needs to be gone through before the extraction of the master signature.

The last objective is to setup a method of analysing the patterns obtained so that the operating conditions can be reflected. A two-step reasoning approach has been constructed. Primarily, the RI chart should be consulted as any deviation from zero

value subject signals an abnormal condition. However, it should be borne in mind that the waiting time for the coming element should not exceed the maximum amplitude obtained from the master signature. If the RI chart cannot show the condition in detail, then the ILT chart can be consulted as two reasoning trees, one for the RI chart and one for the ILT chart, have been built.

In terms of experiments, a software program has been coded in which the use of DLL caters for better portability and this makes the transformation to another platform easy. Test cases involving Overflow, Normal, Slowdown, Blocking, and Unknown Event Occurrence were studied. It was observed that in our experiments that the proposed methodology was able to identify most conditions of the element flow. This research makes a contribution to the field of the monitoring of dynamic systems as the hardware requirements are very simple. This approach is comparable to the ANDON system in that it is a good generic method suitable for most production systems. In fact, the human intervention factor can be reduced in the proposed approach. And to a certain extent, the decision make/reasoning part has the potential to be of use in the field of automation.

To further enhance the deterministic algorithms of the proposed approach, the cyclical characteristics of an event can be modified in terms of periodical manner. By doing so, the deterministic results can be more practical for many applications. Moreover, apply sophisticated pattern recognizing methods which have been already developed by scholars and researchers to identify the conditions of the ROIs can improve the efficiency of the system reasoning ability. These can be the directions for scholars who are interested to carry further research studies.

References

Ahuja, R., Magnanti, T. and Orlin, J. (1993), *Network Flows Theroy, Algorithms and Applications, Prentice Hall*, Upper Saddle River, New Jersey.

Benton, W. and Shin, H. (1998), 'Manufacturing Planning and Control: The evolution of MRP and JIT Integration', *European Journal of Operational Research*, Vol.1, No.10, pp.411-440.

Calvin, R. (2002), *Entrepreneurial management*, McGraw-Hill, New York.

Chen, J. (1994), 'In-Process Pokayoke System in Unmanned Manufacturing Cells', *Proceedings of IEEE International Conference on Industrial Technology*, Dec, Guangzhou, China, pp.345-349.

Cheng, J. and Chou, C. (2008), 'A Real-Time Inventory Decision System Using Western Electric Run Rules and ARMA Control Chart', *Expert Systems with Applications: An International Journal*, Vol.35, No.3, pp.755-761.

Dennis, P. (2002), *Lean Production Simplified: a Plain Language Guide to the World's Most Powerful Production System*, New York: Productivity Press.

Dijkstra, E. (1959), 'A Note on Two Problems in Connexion with Graphs', *Numerische Mathematik*, Vol.1, No.1, pp.269-271.

Dorf, R. and Bishop, R. (1998), *Modern Control Systems*, Eighth Edition, Addison-Wesley.

Fane, G.R., Vaghefi, M.R., Deusen, C.V. and Woods, L.A. (2003), 'Competitive Advantage the Toyota Way', *Business Strategy Review*, Vol.14, No.4, pp.51-60.

Forrester, J. (1958), 'Industrial Dynamics', *Harvard Business Review*, Vol.36, No.4, pp.37-66.

Forrester, J. (1961), *Industrial Dynamics*, The M.I.T. press.

References

Fujimoto, Shun (1997)**,** *Fengtian Shi Guan Li Ben Zhi*, Taibei: Zhongguo Sheng Chan Li Zhong Xin.

Green, E. (2007) 'Empirical Management of Engineering Projects: A Common Sense Approach to Agility', *Agile Manufacturing ICAM IET International Conference*, Durham, UK, 9-11 July 2007, pp. 74-77.

Guh, R. (2005), 'Real-Time Pattern Recognition in Statistical Process Control: A Hybrid Neural Network/Decission Tree-Based Approach', *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, Vo.213, No.3, Mar, pp.283-298.

Guh, R. (2005), 'A Hybrid Learning-Based Model for On-Line Detection and Analysis of Control Chart Patterns', *Computer and Industrial Engineering*, Vol.49, No.1, pp.35-62.

Hagoort, G. (2003), *Art Management: Entrepreneurial Style*, Delft: Eburon Publishers.

Hassan, A., Baksh, M, Shaharoun, A. and Jamaluddin, H. (2003), 'Improved SPC Chart Pattern Recognition Using Statistical Feature', *International Journal of Production Research*, Vol.41, No.7, pp.1587-1603.

Haykin, S. (1999), *Neural Networks, a Comprehensive Foundation*, 2nd Edition, Englewood Cliffs, N.J.: Prentice Hall.

Higgins, P. and Browne, J. (1990), 'The Monitor in Production Activity Control Systems', *Production Planning and Control,* Vol.1, No.1, pp.17-26.

Hwarng, H. and Hubele, N. (1991), ' X-Bar Chart Pattern Recognition Using Neural Nets', *ASQC Quality Congress Transaction*, Vol.45, pp.884-889.

Jain, A., Duin, R. and Mao, J. (2000), 'Statistical Pattern Recognition: A Review', *IEE Transcations on Pattern Analysis Intelligence*, Vol.22, No.1, pp.4- 37.

References

Jia, X., Wan, C. and Song, J. (2001), 'SPC Control Chart for Clustered Deflects', *Research and Progress of SSE*, Vol.21, No.4, pp.425-430.

Kelton, W., Sadowski, R. and Sturrock, D. (2004), *Simulation with Arena*, 3rd Edition. Boston, Mass.: McGraw-Hill Higher Education.

Lewis, R. and Ransing, R. (1997), 'A Semantically Constrained Bayesian Network for Manufacturing Diagnosis', *International Journal of Production Research*, Vol.35, No.8, pp.2171-2187.

Li, J. and Blumenfeld, D. (2005), 'Analysis of ANDON Type Transfer Production Lines: A Quantitative Approach', *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, April, Barcelona, Spain, pp.278-283.

Liu, Q. (1998), 'Pattern Recognition of Machine Tool Faults with a Fuzzy Mathematics Algorithm', *International Journal of Production Research*, Vol.36, No.8, pp.2301-2314.

Mass, W. and Robertson, A. (1996), 'From Textiles to Automobiles: Mechanical and Organizational Innovation in the Toyoda Enterprise, 1895-1933', *Business and Economic History*, Vol.12, No.2, pp.1-37.

McKinney, D. and a Volunteer Group of Contributors within the International Council on Systems Engineering (2004), *INCOSE Systems Engineering Handbook*, Version 2a, INCOSE, Indiana.

Mendonca, J. M., Ribeiro, B., Schulte, J., Weller, R. and Santos, J. M. (1996) 'Building User-Driven solutions for Decision-Making in Manufacturing', *IT and Manufacturing Partnerships Delivering the promise*, Galway, Ireland, October, pp. 239-251.

Monden, Yasuhira (1998), *Toyota Production System*, Third Edition, Industrial

Engineering & Management Press, Atlanta, Georgia, USA.

Murdoch, J. (1979), *Control Charts*, Palgrave Macmillan, London.

Nieman, H. (1990), *Pattern Analysis and Understanding 2nd Edition*, Heidelberg: Springer-Verlag.

Ono, Taiichi (1988), *Toyota Production System: Beyond Large-Scale Production*, Cambridge, Mass.: Productivity Press.

Ostle, B. (1996), *Engineering Statistics: The Industrial Experience*, Belmont, Calif.: Duxbury Press.

Ovacik, I. and Uzsoy, R. (1993), 'Exploiting Real-Time Shop Floor Status Information to Schedule Complex Job Shops', *Proceedings of the Industrial Engineering Research Conference*, CA, USA, pp.868-872.

Rauwendaal, C. (2000), *SPC: Statistical Process Control in Injection Molding and Extrusion*, Munich: Hanser Publishers.

Robert R. Inman, Dennis E. Blumenfeld, Ningjian Huang and Jingshan Li (2003) 'Designing Production Systems for Quality: Research Opportunities from an Automotive Industry Perspective', *International Journal of Production Research*, Vol.41, No.9, pp.1953 - 1971

Stevenson, W. (2002), *Operations Management*, McGraw_Hill, Irwin.

Sun, W. (1994) 'The Dynamic Simulation Model of Industrial Enterprise Systems and its Application', *Journal of Systems Science and Systems Engineering*, Vol. 3, No. 2, pp. 153-156

Szelk, E. and Meszaros, I. (1990), 'Intelligent and Adaptive Control of FMS Contributing to CIM Trend', *IFAC 11th Triennial World Congress*, Tallinn Estonia, USSR, pp.31-36.

Taylor, F. (2003), *Scientific Management*, Routledge, London ; New York.

References

Tinham, B. (2005), 'IT Drives Toyota's Lean Improvement', *Manufacturing Computer Solutions*, Vol.11, No.3, pp.45.

Toh, K. and Devanathan, R. (1993), 'Process Identification by Neuromorphic Pattern Recognition', *IECON Proceedings (Industrial Electronics Conference)*, Vol.1, Nov., Maui, Hawaii, USA, pp.349-353.

Toh, K. and Devanathan, R. (1994), 'Pattern Recognition in Process Control: An Instrumental Variable Approach', *Proceedings of 1994 IEEE Region 10's Ninth Annual International Conference Theme: Fronties of Computer Technology*, Vol.2, No.2, Aug, Singapore, pp.966-970.

Tseng, H., Ip., W. and Ng, K. (1999), 'A Model for an Integrated Manufacturing System Implementation in China: a Case Study', *Journal of Engineering and Technology Management*, Vol.16, No.1, pp. 83-101.

Walpole, R. (1993), *Probability and Statistics for Engineers and Scientists*, 5th Edition, New York: Macmillan.

Watanabe, S. (1985), *Pattern Recognition: Human and Mechanical*, New York: Wiley.

West, Douglas Brent (2001), *Introduction to Graph Theory*, 2nd Edition, Upper Saddle River, N.J. : Prentice Hall

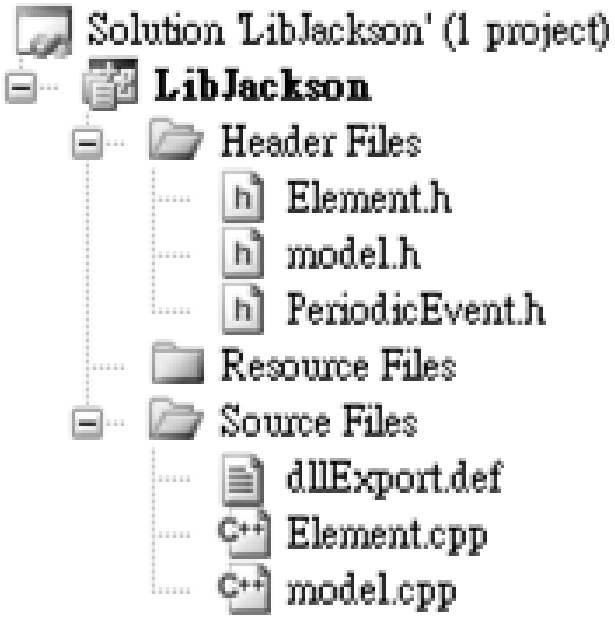
Western Electric Company, (1958), *Statistical quality control handbook*, 2nd Edition**,** New York Publishers.

Yang, J.H. and Yang, M.S., (2005), "A Control Chart Pattern Recognition System Using a Statistical Correlation Coefficient Method", *Computers and Industrial Engineering*, Vol. 48, No. 2, p.p. 205-221.

Zhu, K. (1996), 'Sensor-Based Condition Monitoring and Predictive Maintenance –

an Integrated Intelligent Management Support System', *International Journal of Intelligent Systems in Accounting, Finance and Management*, Vol.5, No.4, pp.241-258.

A. Appendix-A – Program Source Code



Model.cpp

```
#include <malloc.h>

#include "model.h"
#include "PeriodicEvent.h"
#include "Element.h"

PRE_CALL int DLL_CALL model_init(double time_standard, double time_gap, int capacity_element,
int capacity_event)
{
	if (__ELEMENT_NUM >0 && __EVENT_NUM>0)
	{
		model_end();
	}
	__TIME_S = time_standard;
	__TIME_G = time_gap;
	__ELEMENT_CAP = capacity_element;
	__EVENT_CAP = capacity_event;
	__ELEMENT_NUM = 0;		// started from 1
	__EVENT_NUM = 0;		// started from 1
	elements = (Element**)malloc(sizeof(Element*) * __ELEMENT_CAP);
	events = (PeriodicEvent**)malloc(sizeof(PeriodicEvent*) * __EVENT_CAP);

	// set 0-th event as "point in"
	PeriodicEvent* pe = new PeriodicEvent();
	pe->id = 0;
	pe->net_time = 0;
	pe->time_eff = 100;
	pe->time_non_eff = 0;
	pe->phase_delay = 0;
	events[0] = pe;

	return 0;
};

PRE_CALL int DLL_CALL model_end()
{
	for (int i=__ELEMENT_NUM;i>0;i--)
	{
		delete elements[i];
	}
	for (int i=__EVENT_NUM;i>=0;i--)
	{
		delete events[i];
	}
	__ELEMENT_NUM = 0;
	__EVENT_NUM = 0;
	free(elements);
	free(events);
	return 0;
}

PRE_CALL int DLL_CALL new_periodic_event(double non_eff_time_per_period, double
eff_time_per_period, double phase_delay, double net_time_period)
{
	int index = ++__EVENT_NUM;
	PeriodicEvent* pe = new PeriodicEvent();
	pe->id = index;
```

```
	pe->net_time = net_time_period;
	pe->time_eff = eff_time_per_period;
	pe->time_non_eff = non_eff_time_per_period;
	pe->phase_delay = phase_delay;
	events[index] = pe;

	return index;

};

PRE_CALL int DLL_CALL new_element(double coming_time)
{
	int index = ++__ELEMENT_NUM;
	Element* e = new Element();
	e->id = index;
	e->evtCap = __EVENT_CAP;
	e->evtNum = __EVENT_NUM;
	e->pEvents = events;
	e->time_st = coming_time;
	e->time_standard = __TIME_S;
	e->time_gap = __TIME_G;
	if (e->id == 1)
	{
		e->prevElement = 0;
	}
	else
	{
		e->prevElement = elements[index - 1];
	}
	e->delta_time = (double*)malloc(sizeof(double) * (__EVENT_CAP + 1)); // allocate mem for
delay time
	e->clear_time = (double*)malloc(sizeof(double) * (__EVENT_CAP + 1)); // allocate mem for
clear time
	e->delta_time[__EVENT_NUM] = -1;	// set this element "Not Ready"
	e->clear_time[__EVENT_NUM] = -1;	// set this element "Not Ready"

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

	return index;
};

PRE_CALL double DLL_CALL get_exp_out_time(int element_id)
{
	return elements[element_id]->get_time_out();
};

PRE_CALL double DLL_CALL get_delta_time(int element_id, int event_id)
{
	get_exp_out_time(element_id);
	return elements[element_id]->delta_time[event_id];
};

PRE_CALL double DLL_CALL get_clear_time(int element_id, int event_id)
{
	get_exp_out_time(element_id);
	return elements[element_id]->clear_time[event_id];
};
```

Element.cpp

```
#include <malloc.h>
#include <assert.h>

#include "Element.h"
#include "PeriodicEvent.h"

Element::Element()
{
	this->id = -1;
	this->prevElement = 0;
	this->pEvents = 0;
	this->evtCap = 0;
	this->evtNum = 0;
	this->time_st = -1;
	this->time_standard = -1;
	this->time_gap = -1;
	this->delta_time = 0;
};

Element::~Element()
{
	free(this->delta_time);
};

void Element::calculate_delta()
{
	if (prevElement && prevElement->delta_time[evtNum] < 0)
	{
		// if previous element is not calculated OK. we do the previous firstly.
		// this code could run recursively, until meet an OK, or the 1st element
		prevElement->calculate_delta();
	}
	for (int i=0; i <= evtNum; i++)
	{
		// consider the delay from the 1st event to the last one
		this->calculate_event_delta(i);
	}
};

void Element::calculate_event_delta(int e_id)
{
	PeriodicEvent* evt = pEvents[e_id];
	double previous_total_delay = 0;
	if (e_id>1)
	{
		previous_total_delay = this->delta_time[e_id - 1];
	}
	double arrive_time = this->time_st + evt->net_time + previous_total_delay;
	double current_delay = previous_total_delay + evt->wait_time(arrive_time);

	// if it's blocked by previous one
	if (this->id > 1)
	{
	    double block_delay_from_prev = (prevElement->clear_time[e_id] + prevElement->time_st)
- (current_delay + this->time_st);
		if ( block_delay_from_prev >0)
		{
```

```
                // temporarily blocked
                // get the new pass time
                current_delay += block_delay_from_prev;
                current_delay = current_delay + evt->wait_time(pEvents[e_id]->net_time +
current_delay);
          }
     }

     // It must can pass this event now.
     this->delta_time[e_id] = current_delay;

     double clear_gap = time_gap;
     if (this->time_standard < evt->net_time + clear_gap)
     {
          clear_gap = this->time_standard - evt->net_time;
     }
     this->clear_time[e_id] = current_delay + clear_gap;  // set earliest clear time, if next
event block me,
                                                                                  // I can update it
by next codes of next event
     // update previous clear_time
     for (int i=e_id -1; i>=0; i--)
     {
          double event_distance = evt->net_time - pEvents[i]->net_time;
          if (event_distance < this->time_gap)
          {
                // a previous event is shorter than time_gap
                // the increament is the difference of two delays
                clear_time[i] += current_delay - delta_time[i];
          }
          else
          {
                break;
          }
     }
};

double Element::get_time_out()
{
     if(delta_time[evtNum] < 0)
     {
          // not ready
          calculate_delta();
     }
     return time_st + time_standard + delta_time[evtNum];
};
```

dllExport.def

```
; IOCorelation.def : Declares the module parameters for the DLL.

LIBRARY      "LibJackson"

EXPORTS
    ; Explicit exports can go here
model_init
model_end
new_periodic_event
new_element
get_exp_out_time
get_delta_time
get_clear_time
```

PeriodicEvent.h

```
#ifndef PERIODIC_EVENT_H
#define PERIODIC_EVENT_H

class PeriodicEvent
{
public:
     int id;                         // event id
     double time_eff;           // eff time per period (cannot pass)
     double time_non_eff; // non-eff time per period (can pass)
     double phase_delay;        // what time does the first "time_non_eff" come
     double net_time;           // distance from beginning

     PeriodicEvent(){};         // constructor
     virtual ~PeriodicEvent(){};     // destructor
     bool can_pass(double t)
     {
          if (id==0)
          {
               // element can pass "point_in" in any time
               return true;
          }
          // For one period,
          //         if 0=< t < time_non_eff,                              then element can
pass
          //         if time_non_eff =< t < time_eff+time_non_eff,   then cannot
          int passed_periods = (int)((t - phase_delay)/(time_eff+time_non_eff));
          double phase_arrival = (int)(t - phase_delay -
(time_eff+time_non_eff)*passed_periods);
          return phase_arrival < time_non_eff;
     };
     double wait_time(double t)
     {
          // how long to wait
          if (can_pass(t))
          {
               return 0.0;
          }
          int passed_periods = (int)((t - phase_delay)/(time_eff+time_non_eff));
          double phase_arrival = t - phase_delay - (time_eff+time_non_eff)*passed_periods;
          return time_eff + time_non_eff - phase_arrival;
     }
};

#endif
```

model.h

```
/**
This is the interface of this .dll project.
Users may call below functions to run
*/


#ifndef MODEL_DLL_H
#define MODEL_DLL_H
#define EXPORT_DLL

#include "PeriodicEvent.h"
#include "Element.h"

// begin C codes
#ifdef __cplusplus
extern "C" {
#endif

#ifdef EXPORT_DLL
     #define PRE_CALL __declspec(dllexport)
     #define DLL_CALL __stdcall
#else
     #define DLL_CALL __declspec(dllimport)
#endif

     PRE_CALL int DLL_CALL  model_init(double time_standard, double time_gap, int
capacity_element, int capacity_event);
     /** initialize the process, especially, prepare memory
     @param:
          time_standard        (T_s) Ideal time to travel from Point_in to Point_out
          time_gap             (T_g) how far should an element keep away from precedent element
          capacity_element     how many elements during this run? (program capacity)
          capacity_event       .........events.................(program capacity)
     @return
          0:         OK
          1:         fail
     */

     PRE_CALL int DLL_CALL model_end();
     /** free the memory
     @return
          0:         OK
     */

     PRE_CALL int DLL_CALL new_periodic_event(double non_eff_time_per_period, double
eff_time_per_period, double phase_delay, double net_time_period);
     /** set up a Boolean periodic event, with (period = non_eff_time_per_period +
eff_time_per_period),
          during eff_time, elements CAN NOT pass; during non_eff_time, elements CAN pass.

          [please insert all events before "model_run()" function, or you need "model_reset()"
to redo all the elements]
     @param
          non_eff_time_per_period:   how long in a period it does NOT affect elements
          eff_time_per_period:       .......................does affect ....
          phase_delay:                    what time is the first non_eff_time starts
                                               see http://en.wikipedia.org/wiki/Group_delay
```

```
for more
          net_time_period:               (T_a) how long does an element need to travel here,
from the beginning
     @return
          x:         x>=0 is the event_id (range: 0 - max_event_num)
          -1:        failed (out of bound) or (program has started)
     */

     PRE_CALL int DLL_CALL new_element(double coming_time);
     /** set up a newly coming element at the Point_in
     @param:
          coming_time          time to come
     @return
          x:         x>=0 is the element_id (range: 0 - max_element_num)
          -1:        fail (out of bound)
     */

     //int DLL_CALL model_run();
     /** run the model to calculate element time
     @return
          0:         run successfully
     */

     PRE_CALL double DLL_CALL get_exp_out_time(int element_id);
     /** return an element's outgoing time
     @param:
          element_id:          index of elements ([0,1,....n-1])
     @return:
          x.xx:      x.xx is the expected outgoing time of given element.
     */

     PRE_CALL void DLL_CALL model_reset();
     /** re-calculate time for all elements
     */

     //
     //int new_nonperiodic_interference(double eff_time, double phase_lag);
     //todo: interference.

     // below 2 are for debug
     PRE_CALL double DLL_CALL get_delta_time(int element_id, int event_id);
     PRE_CALL double DLL_CALL get_clear_time(int element_id, int event_id);

#ifdef __cplusplus
}
#endif     // end of C codes


// local variables to use
static int __ELEMENT_CAP = 0;
static int __EVENT_CAP = 0;
static int __ELEMENT_NUM = 0;
static int __EVENT_NUM = 0;
static double __TIME_S = 0;
static double __TIME_G = 0;
PeriodicEvent** events;
Element** elements;

#endif  // MODEL_DLL_H
```

Element.h

```
#include "PeriodicEvent.h"

#ifndef ELEMENT_H
#define ELEMENT_H

class Element
{
public:
     int id;
     Element* prevElement;      // previous element
     PeriodicEvent** pEvents;   // poiner to PEvent pointer
     int evtCap;
     int evtNum;


     double time_st;                 // start time
     double time_standard;      // standard time;
     double time_gap;                // gap time
     double* delta_time;             // array of time (totally) delayed (itself) after passing
n-th event
     double* clear_time;             // array of clear (non-clocked any more) time for passed
events
     // above two arrays are started from 0, i.e [0,1,...,n]
     // but the 0-th is the "point_in", 1-st is the event No.1.

     Element();                      // constructor
     virtual ~Element();             // destructor
     void calculate_delta();
     /** a func to work out all the delta, by calling "calculate_event_delta(int)" one by one.
     */
     void calculate_event_delta(int event_id);
     /** for one event ..... dependent on the previous event.
     */
     double get_time_out();
     /** return what we want
     @return
          x.xx       x.xx is the exp time,
                          x.xx = time_standard + total delay after last event
     */
};

#endif
```

B. Appendix-B – Hand Simulation

A ROI configuration setting has been prepared to evaluate the proposed software program, see **Chapter 4** for the program construction. The configuration contains three events and those contain no discrete distributions. The settings are shown in **Table B–1**. The purpose of this section is to compare the computational results obtained from the program and the results calculated by hand. The calculation procedure is in accordance with the theory developed which is described in **Chapter 3**.

The arrival time (T_e) of the incoming element is at a point which is 5 time units past the starting point, and the clock time of when the first element arrived at the inlet is also at a point of 5 time units (minutes for example) after the starting point. Therefore, the values of the elements input time ($\boldsymbol{t_{i,\beta}}$) are 5, 10 and 15 time units. A program calculation result is summarized at the last column of **Table B–2**. The computational outcome of the program is shown by a square box composed of dotted lines in **Figure B–1**.

Table B-1. ROI Configuration and Event Cycle Definitions

ROI Configuration	
T_s	10
T_g	1
Event Cycle Definitions	

k	$T_{k,pc}$	$T_{k,po}$	T_k	$t_{k,p}$	$t_{k,p,m}$	$P(t_{k,p,m})$
1	4	2	2	0	---	1
2	5	4	5	1	---	1
3	2	3	6	2	---	1

The Calculation of Element Leaving Time

The 1st element that enters *(arrival time = 5 time units after the starting time)*

The first element arrived at the inlet ($t_{i,\beta}$) at the clock time of 5 time units (minutes for example) past the starting point. The possible current element arrival time ($t_{1,\beta}$) to the event 1 can be calculated by **Equation 1 from Chapter 3**.

$$t_{1,\beta} = t_{i,\beta} + \check{T}_{1,\beta}$$

$$= 5 + 2$$

$$= 7 \text{ time unit}$$

In the expression ($\check{T}_{1,\beta}$) , the value is equal to (T_1) since it is the first element to enter the first event of the ROI, see **Equation 4**. On the other hand, it is necessary to calculate the relative values through **Equation 2 and 3**. By using **Equation 3**, the deterministic extended time of current element caused by the effective time of the first event, ($\Delta T_{1,\beta}$) can be calculated. The value of the positional order of the first event (n_1) is required for the **Equation 3** calculation. (n_1) can be found by **Equation 2**.

$$t_{1,p} + n_1 \cdot T_{1,p} \le t_{1,\beta} < t_{1,p} + (n_1 + 1) \cdot T_{1,p}$$

$$6n_1 \le 7 < (n_1 + 1) \cdot 6$$

try $n_1 = 0$ $\quad 0 \le 7 < 6 \quad$ *(not valid)*

try $n_1 = 1$ $\quad 6 \le 7 < 12 \quad$ *(true)*

$\therefore n_1 = 1$

Substituting this value into **Equation 3**,

$$\Delta T_{1,\beta} = \max[(t_{1,p} + n_1 \cdot T_{1,p} + T_{1,pc}) - t_{1,\beta} , 0]$$

$$= \max[(0 + 1 \cdot 6 + 4) - 7 , 0]$$

$= 3$ (which is greater than 0),

$\therefore \Delta T_{1,\beta} = 3$ time unit.

Substitute ($\Delta T_{1,\beta}$) into **Equation 4**, we get;

$$\breve{T}_{k,\beta} = T_k + \sum_{i=1}^{k-1} \Delta T_{i,\beta}$$

$$\breve{T}_{2,\beta} = 5 + 3$$

$\therefore \breve{T}_{2,\beta} = 8$ time units.

Go back to the calculation loop, from **Equations 1 to 4**. The values of ($\breve{T}_{3,\beta}$), ($t_{2,\beta}$) and ($t_{3,\beta}$) can also be calculated; they are 11, 13 and 16 units of time for the second event of the same element. Repeat the calculation loop until the last event. We can see that the value of ($\breve{T}_{3,\beta}$) is equal to "0" units of time. Based on the above values, the leaving time of the first element can be calculated by **Equation 5**;

$$t_{o,\beta} = t_{i,\beta} + T_s + \sum_{i=1}^{K} \Delta T_{i,\beta}$$

$$= 5 + 10 + 3 + 2 + 0$$

= 20 time units (The leaving time of the 1st element to enter)

The 2nd entry elements *(arrival clock time = 10 time unit)*

The second element arrival clock time at the inlet ($t_{i,\beta}$) is 10 time unit. For the non-first entry elements, the element leaving would be possible influenced by the traffic ahead. This case is necessary to consider the **Equation 6**, and it is required to returns the updated values back to the equations which have been discussed in the "The 1st entry element section".

Step 1, calculating the arrival time to the first event of the second element to enter, by using equation 1;

$$t_{1,\beta} = t_{i,\beta} + \check{T}_{1,\beta}$$

$$= 10 + 2$$

$$= 12 \text{ time unit.}$$

Step 2, using **Equation 6** to check whether the previous element "β-1" holds off the second element;

$$t_{k,\beta} = \max[\ (t_{k,\beta-1} + \breve{T}_{k,\beta-1} + T_g)\ ,\ t_{k,\beta}\ ]$$

$$t_{1,\beta} = \max[\ (t_{1,\beta-1} + \check{T}_{1,\beta-1} + T_g)\ ,\ t_{1,\beta}]$$

$$= \max[\ 11, 12\ ]$$

$$= 12 \text{ time unit.}$$

When the clock time for the second element arrives to the first event is later than the element leaving time of the first element, the first element does not hold the second element. So that, the deterministic arrival time to the first event of the second element is 12min.

Step 3, use the updated $t_{1,\beta}$ and feeding back to **Equations 1 and 4** to calculate the $(\Delta T_{1,\beta})$ and $(\check{T}_{2,\beta})$; and they are "4" and "9" time unit.

By repeating step 1 to 3, get the other related values same as the "The first entry element". Finally, we can calculate the $(\check{T}_{s,\beta})$ by **Equation 4**;

$$\check{T}_{s,\beta} = 10 + 4 + 5 + 0 + 0 = 19 \text{ time unit.}$$

Eventually, the element leaving time of the second element to enter can be calculated by **Equation 5**;

$t_{o,\beta} = 10 + 19$

$= 29$ time unit (The leaving time of the 2nd entry element)

The 3rd entry elements *(arrival clock time = 15 unit time)*

By using the same calculation steps as the 2nd entry element, the deterministic element leaving time of the third element is 30 unit time.

The same set of parameters and element entry time has been run by the software program which developed in **Chapter 4**, see the dotted line square box in **Figure B–1**. The results are listed in **Table B–2**. It can be seen that the program is running as expected.

Table B-2. Element leaving time summary

			$t_{o,\beta}$	
Element	$t_{i,\beta}$	$P(t_{k,p,i})$	**By hand calculation**	**By program**
1	5	1	20	20
2	10	1	29	29
3	15	1	30	30

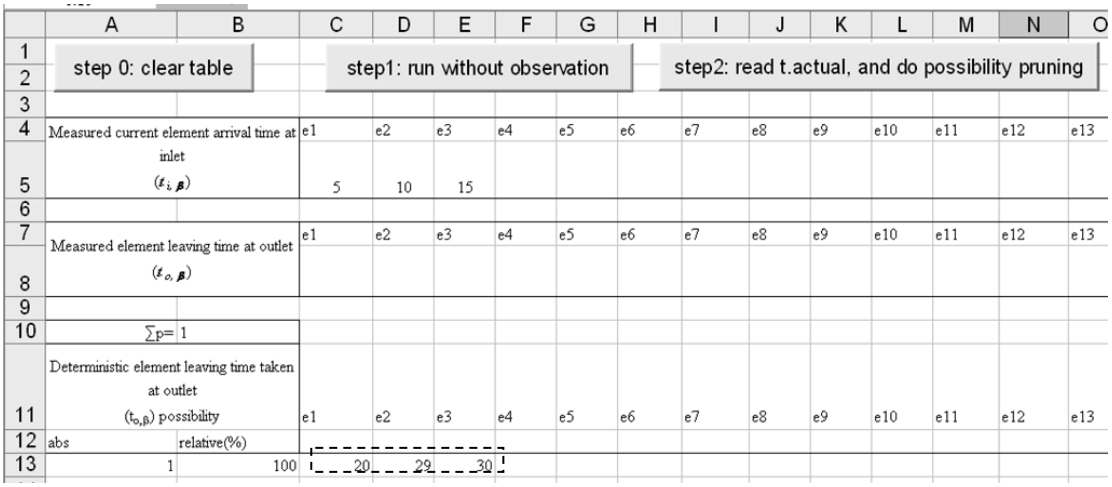


Figure B-1. The Results from Running the Program

C. Appendix-C – Supplementary Table

Table C-1. Example of ROI Configuration

Ts=	380	Time to travel all the distance without any events.
Tg=	1	Distantly equivalent travel time,
cap element=	300	for a constant element travelling gap.
cap events=	10	

Table C-2. Example of Event Setting

event_id	effective_time	non_effective_time	net_time	
1	10	5	70	Comments: An accident event,
2	8	5	220	is equal to a "non_eff=100000"
3	12	8	330	periodic event.
				Event num <= 10

Table C-3. Example of Discrete Time-Slot Distributions

event_id	phase_delay	possibility
1	5	0.2
	6	0.5
	7	0.2
	8	0.1
2	5	0.3
	6	0.5
	7	0.2
3	2	0.25
	3	0.5
	4	0.25

Comments: usually, the sum of possibilities for one event is 1.0. But it's not necessary. 3.3, 0.2, even 0.001 are also acceptable.

∏num(P_ei) <=5000, or Excel 'd crash

Table C-4. Example of ROI Configuration with Discrete Distribution

ROI Configuration					
T_s	10				
T_g	1				
Event Cycles Definition					
k	$T_{k,pc}$	$T_{k,po}$	T_k	$t_{k,p,m}$	$P(t_{k,p,i})$
1	4	2	2	0	0.95
				0.1	0.05
2	5	4	5	0	0.75
				-0.5	0.25
3	2	3	6	0	0.8
				-1	0.15
				0.5	0.05

Table C-5. Normal Case Data

Element	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$t_{i,\beta}$	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85
$t_{o,\beta}$	72	74	82	82.1	89	92	98	107	107.1	112	117	127	128	134	137	143	147
$t_{o,\beta}$	72	74	82	82.1	89	92	98	107	107.1	112	117	127	128	134	137	143	147
$t_{o,\beta} - t_{o,(\beta-1)}$	---	2	8	0.1	6.9	3	6	9	0.1	4.9	5	10	1	6	3	6	4

Element	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
$t_{i,\beta}$	90	95	100	105	110	115	120	125	130	135	140	145	150	155	160	165	170
$t_{o,\beta}$	152	162	164	172	172.1	179	182	188	197	197.1	202	207	217	218	224	227	233
$t_{o,\beta}$	152	162	164	172	172.1	179	182	188	197	197.1	202	207	217	218	224	227	233
$t_{o,\beta} - t_{o,(\beta-1)}$	5	10	2	8	0.1	6.9	3	6	9	0.1	4.9	5	10	1	6	3	6

Element	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$t_{i,\beta}$	175	180	185	190	195	200	205	210	215	220	225	230	235	240	245	250
$t_{o,\beta}$	237	242	252	254	262	262.1	269	272	278	287	287.1	292	297	307	308	314
$t_{o,\beta}$	237	242	252	254	262	262.1	269	272	278	287	287.1	292	297	307	308	314
$t_{o,\beta} - t_{o,(\beta-1)}$	4	5	10	2	8	0.1	6.9	3	6	9	0.1	4.9	5	10	1	6

Table C-6. Overflow Case Data

Element	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
$t_{i,\beta}$	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95
$t_{o,\beta}$	72	74	82	82.1	89	92	98	107	107.1	112	117	127	128	134	137	143	147	152	162
$t_{o,\beta}$	72	82	89	98	107	117	127	134	143	152	162	172	179	188	197	207	217	224	233
$t_{o,\beta} - t_{o,\beta}$	0	8	7	15.9	18	25	29	27	35.9	40	45	45	51	54	60	64	70	72	71
$t_{o,\beta} - t_{o,(\beta-1)}$	---	10	7	9	9	10	10	7	9	9	10	10	7	9	9	10	10	7	9

Element	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
$t_{i,\beta}$	100	105	110	115	120	125	130	135	140	145	150	155	160	165	170	175	180
$t_{o,\beta}$	164	172	172.1	179	182	188	197	197.1	202	207	217	218	224	227	233	237	242
$t_{o,\beta}$	242	252	262	269	278	287	297	307	314	323	332	342	352	359	368	377	387
$t_{o,\beta} - t_{o,\beta}$	78	80	89.9	90	96	99	100	109.9	112	116	115	124	128	132	135	140	145
$t_{o,\beta} - t_{o,(\beta-1)}$	9	10	10	7	9	9	10	10	7	9	9	10	10	7	9	9	10

Element	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$t_{i,\beta}$	185	190	195	200	205	210	215	220	225	230	235	240	245	250
$t_{o,\beta}$	252	254	262	262.1	269	272	278	287	287.1	292	297	307	308	314
$t_{o,\beta}$	397	404	413	422	432	442	449	458	467	477	487	494	503	512
$t_{o,\beta} - t_{o,\beta}$	145	150	151	159.9	163	170	171	171	179.9	185	190	187	195	198
$t_{o,\beta} - t_{o,(\beta-1)}$	10	7	9	9	10	10	7	9	9	10	10	7	9	9

Table C-7. Slowdown Case Configuration

		Phase Compensation Factor
Ts=	60	3
Tg=	0.1	

k	$t_{k,p}$ or $t_{k,p,m}$	
1	0	1
2	0	1
3	0	1

k	$T_{k,pc}$	$T_{k,po}$	T_k
1	4	2	20
2	5	4	30
3	2	3	50

Slow down setting before 1st event				Phase Compensation Factor
1	4	2	20	3
2	5	4	30	3
3	2	3	50	3

Slow down setting between 1st & 2nd event				Phase Compensation Factor
1	4	2	20	0
2	5	4	30	3
3	2	3	50	3

Slow down setting between 2nd & 3rd event				Phase Compensation Factor
1	4	2	20	0
2	5	4	30	0
3	2	3	50	3

Slow down setting after third event in				Phase Compensation Factor
1	4	2	20	0
2	5	4	30	0
3	2	3	50	0

Table C-8. Slowdown Case Data – Before 1st Event

Element	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$t_{i,\beta}$	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90
$t_{o,\beta}$	72	74	82	82.1	89	92	98	107	107.1	112	117	127	128	134	137	143	147	152
$t_{o,\beta}$	72	74	82	89	92	98	98.1	107	112	117	127	127.1	128	134	143	147	152	153
$t_{o,\beta} - t_{o,\beta}$	0	0	0	6.9	3	6	0.1	0	4.9	5	10	0.1	0	0	6	4	5	1
$t_{o,\beta} - t_{o,(\beta-1)}$	---	2	8	7	3	6	0.1	8.9	5	5	10	0.1	0.9	6	9	4	5	1

Element	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
$t_{i,\beta}$	95	100	105	110	115	120	125	130	135	140	145	150	155	160	165	170
$t_{o,\beta}$	162	164	172	172.1	179	182	188	197	197.1	202	207	217	218	224	227	233
$t_{o,\beta}$	162	164	172	179	182	188	188.1	197	202	207	217	217.1	218	224	233	237
$t_{o,\beta} - t_{o,\beta}$	0	0	0	6.9	3	6	0.1	0	4.9	5	10	0.1	0	0	6	4
$t_{o,\beta} - t_{o,(\beta-1)}$	9	2	8	7	3	6	0.1	8.9	5	5	10	0.1	0.9	6	9	4

Element	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$t_{i,\beta}$	175	180	185	190	195	200	205	210	215	220	225	230	235	240	245	250
$t_{o,\beta}$	237	242	252	254	262	262.1	269	272	278	287	287.1	292	297	307	308	314
$t_{o,\beta}$	242	243	252	254	262	269	272	278	278.1	287	292	297	307	307.1	308	314
$t_{o,\beta} - t_{o,\beta}$	5	1	0	0	0	6.9	3	6	0.1	0	4.9	5	10	0.1	0	0
$t_{o,\beta} - t_{o,(\beta-1)}$	5	1	9	2	8	7	3	6	0.1	8.9	5	5	10	0.1	0.9	6

Table C-9. Slowdown Case Data – Between 1st and 2nd Event

Element	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$t_{i,\beta}$	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80
$t_{o,\beta}$	72	74	82	82.1	89	92	98	107	107.1	112	117	127	128	134	137	143
$t_{o,\beta}$	72	82	82.1	83	89	98	102	107	108	117	119	127	134	137	143	143.1
$t_{o,\beta} - t_{o,\beta}$	0	8	0.1	0.9	0	6	4	0	0.9	5	2	0	6	3	6	0.1
$t_{o,\beta} - t_{o,(\beta-1)}$	---	10	0.1	0.9	6	9	4	5	1	9	2	8	7	3	6	0.1

Element	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
$t_{i,\beta}$	85	90	95	100	105	110	115	120	125	130	135	140	145	150	155	160	165
$t_{o,\beta}$	147	152	162	164	172	172.1	179	182	188	197	197.1	202	207	217	218	224	227
$t_{o,\beta}$	152	157	162	172	172.1	173	179	188	192	197	198	207	209	217	224	227	233
$t_{o,\beta} - t_{o,\beta}$	5	5	0	8	0.1	0.9	0	6	4	0	0.9	5	2	0	6	3	6
$t_{o,\beta} - t_{o,(\beta-1)}$	8.9	5	5	10	0.1	0.9	6	9	4	5	1	9	2	8	7	3	6

Element	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$t_{i,\beta}$	170	175	180	185	190	195	200	205	210	215	220	225	230	235	240	245	250
$t_{o,\beta}$	233	237	242	252	254	262	262.1	269	272	278	287	287.1	292	297	307	308	314
$t_{o,\beta}$	233.1	242	247	252	262	262.1	263	269	278	282	287	288	297	299	307	314	317
$t_{o,\beta} - t_{o,\beta}$	0.1	5	5	0	8	0.1	0.9	0	6	4	0	0.9	5	2	0	6	3
$t_{o,\beta} - t_{o,(\beta-1)}$	0.1	8.9	5	5	10	0.1	0.9	6	9	4	5	1	9	2	8	7	3

Table C-10. Slowdown Case Data – Between 2nd and 3rd Event

Element	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$t_{i,\beta}$	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80
$t_{o,\beta}$	72	74	82	82.1	89	92	98	107	107.1	112	117	127	128	134	137	143
$t_{o,\beta}$	74	77	83	83.1	92	97	102	112	112.1	113	119	128	132	137	138	147
$t_{o,\beta} - t_{o,\beta}$	2	3	1	1	3	5	4	5	5	1	2	1	4	3	1	4
$t_{o,\beta} - t_{o,(\beta-1)}$	---	3	6	0.1	8.9	5	5	10	0.1	0.9	6	9	4	5	1	9

Element	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
$t_{i,\beta}$	85	90	95	100	105	110	115	120	125	130	135	140	145	150	155	160	165
$t_{o,\beta}$	147	152	162	164	172	172.1	179	182	188	197	197.1	202	207	217	218	224	227
$t_{o,\beta}$	149	157	164	167	173	173.1	182	187	192	202	202.1	203	209	218	222	227	228
$t_{o,\beta} - t_{o,\beta}$	2	5	2	3	1	1	3	5	4	5	5	1	2	1	4	3	1
$t_{o,\beta} - t_{o,(\beta-1)}$	2	8	7	3	6	0.1	8.9	5	5	10	0.1	0.9	6	9	4	5	1

Element	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$t_{i,\beta}$	170	175	180	185	190	195	200	205	210	215	220	225	230	235	240	245	250
$t_{o,\beta}$	233	237	242	252	254	262	262.1	269	272	278	287	287.1	292	297	307	308	314
$t_{o,\beta}$	237	239	247	254	257	263	263.1	272	277	282	292	292.1	293	299	308	312	317
$t_{o,\beta} - t_{o,\beta}$	4	2	5	2	3	1	1	3	5	4	5	5	1	2	1	4	3
$t_{o,\beta} - t_{o,(\beta-1)}$	9	2	8	7	3	6	0.1	8.9	5	5	10	0.1	0.9	6	9	4	5

Table C-11. Slowdown Case Data – After 3rd Event

Element	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$t_{i,\beta}$	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80
$t_{o,\beta}$	72	74	82	82.1	89	92	98	107	107.1	112	117	127	128	134	137	143
$t_{o,\beta}$	75	77	85	85.1	92	95	101	110	110.1	115	120	130	131	137	140	146
$t_{o,\beta} - t_{o,\beta}$	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
$t_{o,\beta} - t_{o,(\beta-1)}$	---	2	8	0.1	6.9	3	6	9	0.1	4.9	5	10	1	6	3	6

Element	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
$t_{i,\beta}$	85	90	95	100	105	110	115	120	125	130	135	140	145	150	155	160	165
$t_{o,\beta}$	147	152	162	164	172	172.1	179	182	188	197	197.1	202	207	217	218	224	227
$t_{o,\beta}$	150	155	165	167	175	175.1	182	185	191	200	200.1	205	210	220	221	227	230
$t_{o,\beta} - t_{o,\beta}$	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
$t_{o,\beta} - t_{o,(\beta-1)}$	4	5	10	2	8	0.1	6.9	3	6	9	0.1	4.9	5	10	1	6	3

Element	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$t_{i,\beta}$	170	175	180	185	190	195	200	205	210	215	220	225	230	235	240	245	250
$t_{o,\beta}$	233	237	242	252	254	262	262.1	269	272	278	287	287.1	292	297	307	308	314
$t_{o,\beta}$	236	240	245	255	257	265	265.1	272	275	281	290	290.1	295	300	310	311	317
$t_{o,\beta} - t_{o,\beta}$	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
$t_{o,\beta} - t_{o,(\beta-1)}$	6	4	5	10	2	8	0.1	6.9	3	6	9	0.1	4.9	5	10	1	6

Table C-12. Blocking Case Configuration

Ts=	60
Tg=	0.1

k	$T_{k,pc}$	$T_{k,po}$	T_k
1	4	2	20
2	5	4	30
3	2	3	50

k	$t_{k,p}$ or $t_{k,p,m}$	
1	0	1
2	0	1
3	0	1

Blocking setting before first event

	40	350	17		-270	1
1	4	2	20	1	0	1
2	5	4	30	2	0	1
3	2	3	50	3	0	1

Blocking setting between 1st & 2nd event

1	4	2	20	1	0	1
	40	350	26		-270	1
2	5	4	30	2	0	1
3	2	3	50	3	0	1

Blocking setting between 2nd & 3rd event

1	4	2	20	1	0	1
2	5	4	30	2	0	1
	40	350	41		-270	1
3	2	3	50	3	0	1

Blocking setting after third event

1	4	2	20	1	0	1
2	5	4	30	2	0	1
3	2	3	50	3	0	1
	40	350	58		-270	1

Table C-13. Blocking Case Data – Before 1st Event

Element	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$t_{i,\beta}$	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80
$t_{o,\beta}$	72	74	82	82.1	89	92	98	107	107.1	112	117	127	128	134	137	143
$t_{o,\beta}$	72	74	82	82.1	89	92	98	107	107.1	112	117	127	128	134	137	143
$t_{o,\beta} - t_{o,\beta}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$t_{o,\beta} - t_{o,(\beta-1)}$	---	2	8	0.1	6.9	3	6	9	0.1	4.9	5	10	1	6	3	6

Element	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
$t_{i,\beta}$	85	90	95	100	105	110	115	120	125	130	135	140	145	150	155	160	165
$t_{o,\beta}$	147	152	162	164	172	172.1	179	182	188	197	197.1	202	207	217	218	224	227
$t_{o,\beta}$	147	152	162	164	207	207.1	207.2	207.3	207.4	207.5	207.6	207.7	207.8	217	218	224	227
$t_{o,\beta} - t_{o,\beta}$	0	0	0	0	35	35	28.2	25.3	19.4	10.5	10.5	5.7	0.8	0	0	0	0
$t_{o,\beta} - t_{o,(\beta-1)}$	4	5	10	2	43	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	9.2	1	6	3

Element	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$t_{i,\beta}$	170	175	180	185	190	195	200	205	210	215	220	225	230	235	240	245	250
$t_{o,\beta}$	233	237	242	252	254	262	262.1	269	272	278	287	287.1	292	297	307	308	314
$t_{o,\beta}$	233	237	242	252	254	262	262.1	269	272	278	287	287.1	292	297	307	308	314
$t_{o,\beta} - t_{o,\beta}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$t_{o,\beta} - t_{o,(\beta-1)}$	6	4	5	10	2	8	0.1	6.9	3	6	9	0.1	4.9	5	10	1	6

Table C-14. Blocking Case Data – Between 1st and 2nd Event

Element	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$t_{i,\beta}$	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80
$t_{o,\beta}$	72	74	82	82.1	89	92	98	107	107.1	112	117	127	128	134	137	143
$t_{o,\beta}$	72	74	82	82.1	89	92	98	107	107.1	112	117	127	128	134	137	143
$t_{o,\beta} - t_{o,\beta}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$t_{o,\beta} - t_{o,(\beta-1)}$	---	2	8	0.1	6.9	3	6	9	0.1	4.9	5	10	1	6	3	6

Element	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
$t_{i,\beta}$	85	90	95	100	105	110	115	120	125	130	135	140	145	150	155	160	165
$t_{o,\beta}$	147	152	162	164	172	172.1	179	182	188	197	197.1	202	207	217	218	224	227
$t_{o,\beta}$	147	152	197	197.1	197.2	197.3	197.4	197.5	197.6	197.7	197.8	202	207	217	218	224	227
$t_{o,\beta} - t_{o,\beta}$	0	0	35	33.1	25.2	25.2	18.4	15.5	9.6	0.7	0.7	0	0	0	0	0	0
$t_{o,\beta} - t_{o,(\beta-1)}$	4	5	45	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	4.2	5	10	1	6	3

Element	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$t_{i,\beta}$	170	175	180	185	190	195	200	205	210	215	220	225	230	235	240	245	250
$t_{o,\beta}$	233	237	242	252	254	262	262.1	269	272	278	287	287.1	292	297	307	308	314
$t_{o,\beta}$	233	237	242	252	254	262	262.1	269	272	278	287	287.1	292	297	307	308	314
$t_{o,\beta} - t_{o,\beta}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$t_{o,\beta} - t_{o,(\beta-1)}$	6	4	5	10	2	8	0.1	6.9	3	6	9	0.1	4.9	5	10	1	6

Table C-15. Blocking Case Data – Between 2nd and 3rd Event

Element	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$t_{i,\beta}$	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80
$t_{o,\beta}$	72	74	82	82.1	89	92	98	107	107.1	112	117	127	128	134	137	143
$t_{o,\beta}$	72	74	82	82.1	89	92	98	107	107.1	112	117	127	128	134	137	179
$t_{o,\beta} - t_{o,\beta}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	36
$t_{o,\beta} - t_{o,(\beta-1)}$	---	2	8	0.1	6.9	3	6	9	0.1	4.9	5	10	1	6	3	42

Element	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
$t_{i,\beta}$	85	90	95	100	105	110	115	120	125	130	135	140	145	150	155	160	165
$t_{o,\beta}$	147	152	162	164	172	172.1	179	182	188	197	197.1	202	207	217	218	224	227
$t_{o,\beta}$	179.1	179.2	179.3	179.4	179.5	179.6	179.7	182	188	197	197.1	202	207	217	218	224	227
$t_{o,\beta} - t_{o,\beta}$	32.1	27.2	17.3	15.4	7.5	7.5	0.7	0	0	0	0	0	0	0	0	0	0
$t_{o,\beta} - t_{o,(\beta-1)}$	0.1	0.1	0.1	0.1	0.1	0.1	0.1	2.3	6	9	0.1	4.9	5	10	1	6	3

Element	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$t_{i,\beta}$	170	175	180	185	190	195	200	205	210	215	220	225	230	235	240	245	250
$t_{o,\beta}$	233	237	242	252	254	262	262.1	269	272	278	287	287.1	292	297	307	308	314
$t_{o,\beta}$	233	237	242	252	254	262	262.1	269	272	278	287	287.1	292	297	307	308	314
$t_{o,\beta} - t_{o,\beta}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$t_{o,\beta} - t_{o,(\beta-1)}$	6	4	5	10	2	8	0.1	6.9	3	6	9	0.1	4.9	5	10	1	6

Table C-16. Blocking Case Data – After 3rd Event

Element	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$t_{i,\beta}$	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80
$t_{o,\beta}$	72	74	82	82.1	89	92	98	107	107.1	112	117	127	128	134	137	143
$t_{o,\beta}$	72	74	82	82.1	89	92	98	107	107.1	112	117	162	162.1	162.2	162.3	162.4
$t_{o,\beta} - t_{o,\beta}$	0	0	0	0	0	0	0	0	0	0	0	35	34.1	28.2	25.3	19.4
$t_{o,\beta} - t_{o,(\beta-1)}$	---	2	8	0.1	6.9	3	6	9	0.1	4.9	5	45	0.1	0.1	0.1	0.1

Element	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
$t_{i,\beta}$	85	90	95	100	105	110	115	120	125	130	135	140	145	150	155	160	165
$t_{o,\beta}$	147	152	162	164	172	172.1	179	182	188	197	197.1	202	207	217	218	224	227
$t_{o,\beta}$	162.5	162.6	162.7	164	172	172.1	179	182	188	197	197.1	202	207	217	218	224	227
$t_{o,\beta} - t_{o,\beta}$	15.5	10.6	0.7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$t_{o,\beta} - t_{o,(\beta-1)}$	0.1	0.1	0.1	1.3	8	0.1	6.9	3	6	9	0.1	4.9	5	10	1	6	3

Element	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$t_{i,\beta}$	170	175	180	185	190	195	200	205	210	215	220	225	230	235	240	245	250
$t_{o,\beta}$	233	237	242	252	254	262	262.1	269	272	278	287	287.1	292	297	307	308	314
$t_{o,\beta}$	233	237	242	252	254	262	262.1	269	272	278	287	287.1	292	297	307	308	314
$t_{o,\beta} - t_{o,\beta}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$t_{o,\beta} - t_{o,(\beta-1)}$	6	4	5	10	2	8	0.1	6.9	3	6	9	0.1	4.9	5	10	1	6

Table C-17. Unknown Event Occurrence Case Configuration

T_S=	60
T_g=	0.1

k	$T_{k,pc}$	$T_{k,po}$	T_k	$t_{k,p}$ or $t_{k,p,m}$
1	4	2	20	1 0 1
2	5	4	30	2 0 1
3	2	3	50	3 0 1
Blocking setting before first event				
	2	2	17	0 1
1	4	2	20	1 0 1
2	5	4	30	2 0 1
3	2	3	50	3 0 1
Blocking setting between 1st & 2nd event				
1	4	2	20	1 0 1
	2	2	26	0 1
2	5	4	30	2 0 1
3	2	3	50	3 0 1
Blocking setting between 2nd & 3rd event				
1	4	2	20	1 0 1
2	5	4	30	2 0 1
	2	2	41	0 1
3	2	3	50	3 0 1
Blocking setting after third event				
1	4	2	20	1 0 1
2	5	4	30	2 0 1
3	2	3	50	3 0 1
	2	2	58	0 1

Table C-18. Unknown Event Occurrence Case Data – Before 1st Event

Element	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$t_{i,\beta}$	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80
$t_{o,\beta}$	72	74	82	82.1	89	92	98	107	107.1	112	117	127	128	134	137	143
$t_{o,\beta}$	72	74	82	82.1	89	92	98	107	107.1	112	117	127	128	134	143	143.1
$t_{o,\beta} - t_{o,\beta}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0.1
$t_{o,\beta} - t_{o,(\beta-1)}$	---	2	8	0.1	6.9	3	6	9	0.1	4.9	5	10	1	6	9	0.1

Element	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
$t_{i,\beta}$	85	90	95	100	105	110	115	120	125	130	135	140	145	150	155	160	165
$t_{o,\beta}$	147	152	162	164	172	172.1	179	182	188	197	197.1	202	207	217	218	224	227
$t_{o,\beta}$	147	152	162	164	172	172.1	179	182	188	197	202	207	207.1	217	218	224	227
$t_{o,\beta} - t_{o,\beta}$	0	0	0	0	0	0	0	0	0	0	4.9	5	0.1	0	0	0	0
$t_{o,\beta} - t_{o,(\beta-1)}$	3.9	5	10	2	8	0.1	6.9	3	6	9	5	5	0.1	9.9	1	6	3

Element	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$t_{i,\beta}$	170	175	180	185	190	195	200	205	210	215	220	225	230	235	240	245	250
$t_{o,\beta}$	233	237	242	252	254	262	262.1	269	272	278	287	287.1	292	297	307	308	314
$t_{o,\beta}$	233	242	242.1	252	254	262	262.1	269	272	278	287	287.1	292	297	307	308	314
$t_{o,\beta} - t_{o,\beta}$	0	5	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$t_{o,\beta} - t_{o,(\beta-1)}$	6	9	0.1	9.9	2	8	0.1	6.9	3	6	9	0.1	4.9	5	10	1	6

Table C-19. Unknown Event Occurrence Case Data –Between 1st and 2nd Event

Element	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$t_{i,\beta}$	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80
$t_{o,\beta}$	72	74	82	82.1	89	92	98	107	107.1	112	117	127	128	134	137	143
$t_{o,\beta}$	72	82	82.1	82.2	89	92	102	107	107.1	117	117.1	127	128	137	137.1	143
$t_{o,\beta} - t_{o,\beta}$	0	8	0.1	0.1	0	0	4	0	0	5	0.1	0	0	3	0.1	0
$t_{o,\beta} - t_{o,(\beta-1)}$	---	10	0.1	0.1	6.8	3	10	5	0.1	9.9	0.1	9.9	1	9	0.1	5.9

Element	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
$t_{i,\beta}$	85	90	95	100	105	110	115	120	125	130	135	140	145	150	155	160	165
$t_{o,\beta}$	147	152	162	164	172	172.1	179	182	188	197	197.1	202	207	217	218	224	227
$t_{o,\beta}$	152	152.1	162	164	172	172.1	179	188	188.1	197	197.1	202	208	217	224	224.1	227
$t_{o,\beta} - t_{o,\beta}$	5	0.1	0	0	0	0	0	6	0.1	0	0	0	1	0	6	0.1	0
$t_{o,\beta} - t_{o,(\beta-1)}$	9	0.1	9.9	2	8	0.1	6.9	9	0.1	8.9	0.1	4.9	6	9	7	0.1	2.9

Element	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$t_{i,\beta}$	170	175	180	185	190	195	200	205	210	215	220	225	230	235	240	245	250
$t_{o,\beta}$	233	237	242	252	254	262	262.1	269	272	278	287	287.1	292	297	307	308	314
$t_{o,\beta}$	233	237	244	252	262	262.1	262.2	269	272	282	287	287.1	297	297.1	307	308	317
$t_{o,\beta} - t_{o,\beta}$	0	0	2	0	8	0.1	0.1	0	0	4	0	0	5	0.1	0	0	3
$t_{o,\beta} - t_{o,(\beta-1)}$	6	4	7	8	10	0.1	0.1	6.8	3	10	5	0.1	9.9	0.1	9.9	1	9

Table C-20. Unknown Event Occurrence Case Data –Between 2nd and 3rd Event

Element	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$t_{i,\beta}$	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80
$t_{o,\beta}$	72	74	82	82.1	89	92	98	107	107.1	112	117	127	128	134	137	143
$t_{o,\beta}$	73	74	82	82.1	89	93	98	109	109.1	112	117	127	129	134	137	147
$t_{o,\beta} - t_{o,\beta}$	1	0	0	0	0	1	0	2	2	0	0	0	1	0	0	4
$t_{o,\beta} - t_{o,(\beta-1)}$	---	1	8	0.1	6.9	4	5	11	0.1	2.9	5	10	2	5	3	10

Element	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
$t_{i,\beta}$	85	90	95	100	105	110	115	120	125	130	135	140	145	150	155	160	165
$t_{o,\beta}$	147	152	162	164	172	172.1	179	182	188	197	197.1	202	207	217	218	224	227
$t_{o,\beta}$	147.1	153	162	167	172	172.1	182	182.1	189	197	197.1	202	207	217	218	227	227.1
$t_{o,\beta} - t_{o,\beta}$	0.1	1	0	3	0	0	3	0.1	1	0	0	0	0	0	0	3	0.1
$t_{o,\beta} - t_{o,(\beta-1)}$	0.1	5.9	9	5	5	0.1	9.9	0.1	6.9	8	0.1	4.9	5	10	1	9	0.1

Element	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$t_{i,\beta}$	170	175	180	185	190	195	200	205	210	215	220	225	230	235	240	245	250
$t_{o,\beta}$	233	237	242	252	254	262	262.1	269	272	278	287	287.1	292	297	307	308	314
$t_{o,\beta}$	233	237	242	253	254	262	262.1	269	273	278	289	289.1	292	297	307	309	314
$t_{o,\beta} - t_{o,\beta}$	0	0	0	1	0	0	0	0	1	0	2	2	0	0	0	1	0
$t_{o,\beta} - t_{o,(\beta-1)}$	5.9	4	5	11	1	8	0.1	6.9	4	5	11	0.1	2.9	5	10	2	5

Table C-21. Unknown Event Occurrence Case Data – After 3rd Event

Element	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$t_{i,\beta}$	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80
$t_{o,\beta}$	72	74	82	82.1	89	92	98	107	107.1	112	117	127	128	134	137	143
$t_{o,\beta}$	72	76	84	84.1	89	92	100	108	108.1	112	117	128	128.1	136	137	144
$t_{o,\beta} - t_{o,\beta}$	0	2	2	2	0	0	2	1	1	0	0	1	0.1	2	0	1
$t_{o,\beta} - t_{o,(\beta-1)}$	---	4	8	0.1	4.9	3	8	8	0.1	3.9	5	11	0.1	7.9	1	7

Element	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
$t_{i,\beta}$	85	90	95	100	105	110	115	120	125	130	135	140	145	150	155	160	165
$t_{o,\beta}$	147	152	162	164	172	172.1	179	182	188	197	197.1	202	207	217	218	224	227
$t_{o,\beta}$	148	152	164	164.1	172	172.1	180	184	188	197	197.1	204	208	217	220	224	228
$t_{o,\beta} - t_{o,\beta}$	1	0	2	0.1	0	0	1	2	0	0	0	2	1	0	2	0	1
$t_{o,\beta} - t_{o,(\beta-1)}$	4	4	12	0.1	7.9	0.1	7.9	4	4	9	0.1	6.9	4	9	3	4	4

Element	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$t_{i,\beta}$	170	175	180	185	190	195	200	205	210	215	220	225	230	235	240	245	250
$t_{o,\beta}$	233	237	242	252	254	262	262.1	269	272	278	287	287.1	292	297	307	308	314
$t_{o,\beta}$	233	237	244	252	256	264	264.1	269	272	280	288	288.1	292	297	308	308.1	316
$t_{o,\beta} - t_{o,\beta}$	0	0	2	0	2	2	2	0	0	2	1	1	0	0	1	0.1	2
$t_{o,\beta} - t_{o,(\beta-1)}$	5	4	7	8	4	8	0.1	4.9	3	8	8	0.1	3.9	5	11	0.1	7.9

Table C-22. (T_e) With 1% Random Tolerance

T_e: 5

1% tolerance : 0.05

Element	1	2	3	4	5	6	7	8	9	10
Regular	5	10	15	20	25	30	35	40	45	50
Random	0.0323	-0.002	0.0387	-0.03	-0.033	-0.006	0.0386	0.041	0.035	-0.047
1% tolerance	5.03	10.00	15.04	19.97	24.97	29.99	35.04	40.04	45.04	49.95

Element	11	12	13	14	15	16	17	18	19	20
Regular	55	60	65	70	75	80	85	90	95	100
Random	0.0389	0.0094	0.0243	0.0072	0.0039	0.044	-0.016	-0.018	-0.005	0.008
1% tolerance	55.04	60.01	65.02	70.01	75.00	80.04	84.98	89.98	94.99	100.01

Element	21	22	23	24	25	26	27	28	29	30
Regular	105	110	115	120	125	130	135	140	145	150
Random	0.0384	0.0107	0.0297	0.0362	-0.036	-0.02	-0.032	0.049	-0.045	-0.025
1% tolerance	105.04	110.01	115.03	120.04	124.96	129.98	134.97	140.05	144.96	149.98

Element	31	32	33	34	35	36	37	38	39	40
Regular	155	160	165	170	175	180	185	190	195	200
Random	-0.022	0.0052	-0.009	-0.038	0.0161	0.037	0.0232	-0.025	-0.036	-0.009
1% tolerance	154.98	160.01	164.99	169.96	175.02	180.04	185.02	189.98	194.96	199.99

Element	41	42	43	44	45	46	47	48	49	50
Regular	205	210	215	220	225	230	235	240	245	250
Random	-0.013	0.0457	-0.039	0.002	-0.025	0.048	-0.022	0.025	-0.039	-0.013
1% tolerance	204.99	210.05	214.96	220.00	224.97	230.05	234.98	240.03	244.96	249.99

Table C-23. (T_e) With 5% Random Tolerance

T_e: **5**

5% tolerance : **0.25**

Element	1	2	3	4	5	6	7	8	9	10
Regular	5	10	15	20	25	30	35	40	45	50
Random	-0.059	-0.2	0.0482	0.1996	0.1923	0.229	-0.243	-0.046	0.182	-0.181
5% tolerance	4.94	9.80	15.05	20.20	25.19	30.23	34.76	39.95	45.18	49.82

Element	11	12	13	14	15	16	17	18	19	20
Regular	55	60	65	70	75	80	85	90	95	100
Random	-0.127	-0.227	-0.234	-0.168	-0.14	-0.241	-0.107	-0.078	0.027	-0.071
5% tolerance	54.87	59.77	64.77	69.83	74.86	79.76	84.89	89.92	95.03	99.93

Element	21	22	23	24	25	26	27	28	29	30
Regular	105	110	115	120	125	130	135	140	145	150
Random	-0.064	-0.072	0.2052	-0.017	-0.037	-0.098	0.2379	0.153	0.246	-0.122
5% tolerance	104.94	109.93	115.21	119.98	124.96	129.90	135.24	140.15	145.25	149.88

Element	31	32	33	34	35	36	37	38	39	40
Regular	155	160	165	170	175	180	185	190	195	200
Random	0.2258	-0.223	0.1025	0.1583	0.2363	-0.017	-0.1	0.125	-0.074	0.1378
5% tolerance	155.23	159.78	165.10	170.16	175.24	179.98	184.90	190.13	194.93	200.14

Element	41	42	43	44	45	46	47	48	49	50
Regular	205	210	215	220	225	230	235	240	245	250
Random	-0.213	-0.151	-0.218	-0.071	-0.006	0.006	-0.063	0.243	-0.23	-0.135
5% tolerance	204.79	209.85	214.78	219.93	224.99	230.01	234.94	240.24	244.77	249.87

Table C-24. (T_e) With 10% Random Tolerance

T_e: **5**

10% tolerance : **0.5**

Element	1	2	3	4	5	6	7	8	9	10
Regular	5	10	15	20	25	30	35	40	45	50
Random	-0.118	-0.399	0.0965	0.3991	0.3846	0.458	-0.486	-0.093	0.363	-0.361
10% tolerance	4.88	9.60	15.10	20.40	25.38	30.46	34.51	39.91	45.36	49.64

Element	11	12	13	14	15	16	17	18	19	20
Regular	55	60	65	70	75	80	85	90	95	100
Random	-0.255	-0.455	-0.468	-0.336	-0.28	-0.483	-0.215	-0.157	0.054	-0.143
10% tolerance	54.75	59.55	64.53	69.66	74.72	79.52	84.79	89.84	95.05	99.86

Element	21	22	23	24	25	26	27	28	29	30
Regular	105	110	115	120	125	130	135	140	145	150
Random	-0.128	-0.144	0.4103	-0.034	-0.074	-0.196	0.4757	0.307	0.491	-0.244
10% tolerance	104.87	109.86	115.41	119.97	124.93	129.80	135.48	140.31	145.49	149.76

Element	31	32	33	34	35	36	37	38	39	40
Regular	155	160	165	170	175	180	185	190	195	200
Random	0.4517	-0.447	0.205	0.3165	0.4725	-0.034	-0.2	0.25	-0.149	0.2757
10% tolerance	155.45	159.55	165.21	170.32	175.47	179.97	184.80	190.25	194.85	200.28

Element	41	42	43	44	45	46	47	48	49	50
Regular	205	210	215	220	225	230	235	240	245	250
Random	-0.426	-0.302	-0.436	-0.142	-0.013	0.011	-0.127	0.486	-0.459	-0.269
10% tolerance	204.57	209.70	214.56	219.86	224.99	230.01	234.87	240.49	244.54	249.73

Table C-25. Example of Non-constant Input Time – (T_e) With 1% Tolerance

Element	1	2	3	4	5	6	7	8	9	10
$t_{i,\beta}$	5.03	10.00	15.04	19.97	24.97	29.99	35.04	40.04	45.04	49.95
$t_{o,\beta}$	72	72.1	82	82.1	89	92	98	107	107.1	112
Element Increment	1	2	3	4	5	6	7	8	9	10
$t_{o,\beta} - t_{o,(\beta-1)}$	---	0.10	9.90	0.10	6.90	3.00	6.00	9.00	0.10	4.90

Element	11	12	13	14	15	16	17	18	19	20
$t_{i,\beta}$	55.04	60.01	65.02	70.01	75.00	80.04	84.98	89.98	94.99	100.01
$t_{o,\beta}$	117	127	128	134	137	143	147	152	162	164
Element Increment	11	12	13	14	15	16	17	18	19	20
$t_{o,\beta} - t_{o,(\beta-1)}$	5.00	10.00	1.00	6.00	3.00	6.00	4.00	5.00	10.00	2.00

Element	21	22	23	24	25	26	27	28	29	30
$t_{i,\beta}$	105.04	110.01	115.03	120.04	124.96	129.98	134.97	140.05	144.96	149.98
$t_{o,\beta}$	172	172.1	179	182	188	189.98	197	202	207	217
Element Increment	21	22	23	24	25	26	27	28	29	30
$t_{o,\beta} - t_{o,(\beta-1)}$	8.00	0.10	6.90	3.00	6.00	1.98	7.02	5.00	5.00	10.00

Element	31	32	33	34	35	36	37	38	39	40
$t_{i,\beta}$	154.98	160.01	164.99	169.96	175.02	180.04	185.02	189.98	194.96	199.99
$t_{o,\beta}$	218	224	224.99	233	237	242	252	252.1	254.964	262
Element Increment	31	32	33	34	35	36	37	38	39	40
$t_{o,\beta} - t_{o,(\beta-1)}$	1.00	6.00	0.99	8.01	4.00	5.00	10.00	0.10	2.86	7.04

Element	41	42	43	44	45	46	47	48	49	50
$t_{i,\beta}$	204.99	210.05	214.96	220.00	224.97	230.05	234.98	240.03	244.96	249.99
$t_{o,\beta}$	269	272	278	287	287.1	292	297	307	308	314
Element Increment	41	42	43	44	45	46	47	48	49	50
$t_{o,\beta} - t_{o,(\beta-1)}$	7.00	3.00	6.00	9.00	0.10	4.90	5.00	10.00	1.00	6.00

Table C-26. Example of Non-constant Input Time – (T_e) With 5% Tolerance

Element	1	2	3	4	5	6	7	8	9	10
$t_{i,\beta}$	4.94	9.80	15.05	20.20	25.19	30.23	34.76	39.95	45.18	49.82
$t_{o,\beta}$	72.00	72.10	82.00	82.10	89.00	92.00	98.00	99.95	107.00	112.00
Element Increment	1	2	3	4	5	6	7	8	9	10
$t_{o,\beta} - t_{o,(\beta-1)}$	---	0.10	9.90	0.10	6.90	3.00	6.00	1.95	7.05	5.00

Element	11	12	13	14	15	16	17	18	19	20
$t_{i,\beta}$	54.87	59.77	64.77	69.83	74.86	79.76	84.89	89.92	95.03	99.93
$t_{o,\beta}$	117.00	127.00	128.00	134.00	134.86	143.00	147.00	152.00	162.00	162.10
Element Increment	11	12	13	14	15	16	17	18	19	20
$t_{o,\beta} - t_{o,(\beta-1)}$	5.00	10.00	1.00	6.00	0.86	8.14	4.00	5.00	10.00	0.10

Element	21	22	23	24	25	26	27	28	29	30
$t_{i,\beta}$	104.94	109.93	115.21	119.98	124.96	129.90	135.24	140.15	145.25	149.88
$t_{o,\beta}$	164.94	172.00	179.00	182.00	188.00	189.90	197.00	202.00	207.00	217.00
Element Increment	21	22	23	24	25	26	27	28	29	30
$t_{o,\beta} - t_{o,(\beta-1)}$	2.84	7.06	7.00	3.00	6.00	1.90	7.10	5.00	5.00	10.00

Element	31	32	33	34	35	36	37	38	39	40
$t_{i,\beta}$	155.23	159.78	165.10	170.16	175.24	179.98	184.90	190.13	194.93	200.14
$t_{o,\beta}$	218.00	224.00	227.00	233.00	237.00	242.00	252.00	254.00	254.93	262.00
Element Increment	31	32	33	34	35	36	37	38	39	40
$t_{o,\beta} - t_{o,(\beta-1)}$	1.00	6.00	3.00	6.00	4.00	5.00	10.00	2.00	0.93	7.07

Element	41	42	43	44	45	46	47	48	49	50
$t_{i,\beta}$	204.79	209.85	214.78	219.93	224.99	230.01	234.94	240.24	244.77	249.87
$t_{o,\beta}$	269.00	272.00	278.00	279.93	287.00	292.00	297.00	307.00	308.00	314.00
Element Increment	41	42	43	44	45	46	47	48	49	50
$t_{o,\beta} - t_{o,(\beta-1)}$	7.00	3.00	6.00	1.93	7.07	5.00	5.00	10.00	1.00	6.00

Table C-27. Example of Non-constant Input Time – (T_e) With 10% Tolerance

Element	1	2	3	4	5	6	7	8	9	10
$t_{i,\beta}$	4.88	9.60	15.10	20.40	25.38	30.46	34.51	39.91	45.36	49.64
$t_{o,\beta}$	72.00	72.10	82.00	82.10	89.00	92.00	98.00	99.91	107.00	112.00
Element Increment	1	2	3	4	5	6	7	8	9	10
$t_{o,\beta} - t_{o,(\beta-1)}$	---	0.10	9.90	0.10	6.90	3.00	6.00	1.91	7.09	5.00

Element	11	12	13	14	15	16	17	18	19	20
$t_{i,\beta}$	54.75	59.55	64.53	69.66	74.72	79.52	84.79	89.84	95.05	99.86
$t_{o,\beta}$	117.00	127.00	128.00	134.00	134.72	143.00	147.00	152.00	162.00	162.10
Element Increment	11	12	13	14	15	16	17	18	19	20
$t_{o,\beta} - t_{o,(\beta-1)}$	5.00	10.00	1.00	6.00	0.72	8.28	4.00	5.00	10.00	0.10

Element	21	22	23	24	25	26	27	28	29	30
$t_{i,\beta}$	104.87	109.86	115.41	119.97	124.93	129.80	135.48	140.31	145.49	149.76
$t_{o,\beta}$	164.87	172.00	179.00	182.00	188.00	189.80	197.00	202.00	207.00	217.00
Element Increment	21	22	23	24	25	26	27	28	29	30
$t_{o,\beta} - t_{o,(\beta-1)}$	2.77	7.13	7.00	3.00	6.00	1.80	7.20	5.00	5.00	10.00

Element	31	32	33	34	35	36	37	38	39	40
$t_{i,\beta}$	155.45	159.55	165.21	170.32	175.47	179.97	184.80	190.25	194.85	200.28
$t_{o,\beta}$	218.00	224.00	227.00	233.00	237.00	242.00	252.00	254.00	254.85	262.00
Element Increment	31	32	33	34	35	36	37	38	39	40
$t_{o,\beta} - t_{o,(\beta-1)}$	1.00	6.00	3.00	6.00	4.00	5.00	10.00	2.00	0.85	7.15

Element	41	42	43	44	45	46	47	48	49	50
$t_{i,\beta}$	204.57	209.70	214.56	219.86	224.99	230.01	234.87	240.49	244.54	249.73
$t_{o,\beta}$	269.00	272.00	278.00	279.86	287.00	292.00	297.00	307.00	308.00	314.00
Element Increment	41	42	43	44	45	46	47	48	49	50
$t_{o,\beta} - t_{o,(\beta-1)}$	7.00	3.00	6.00	1.86	7.14	5.00	5.00	10.00	1.00	6.00

D. Appendix-D – Supplementary Figure

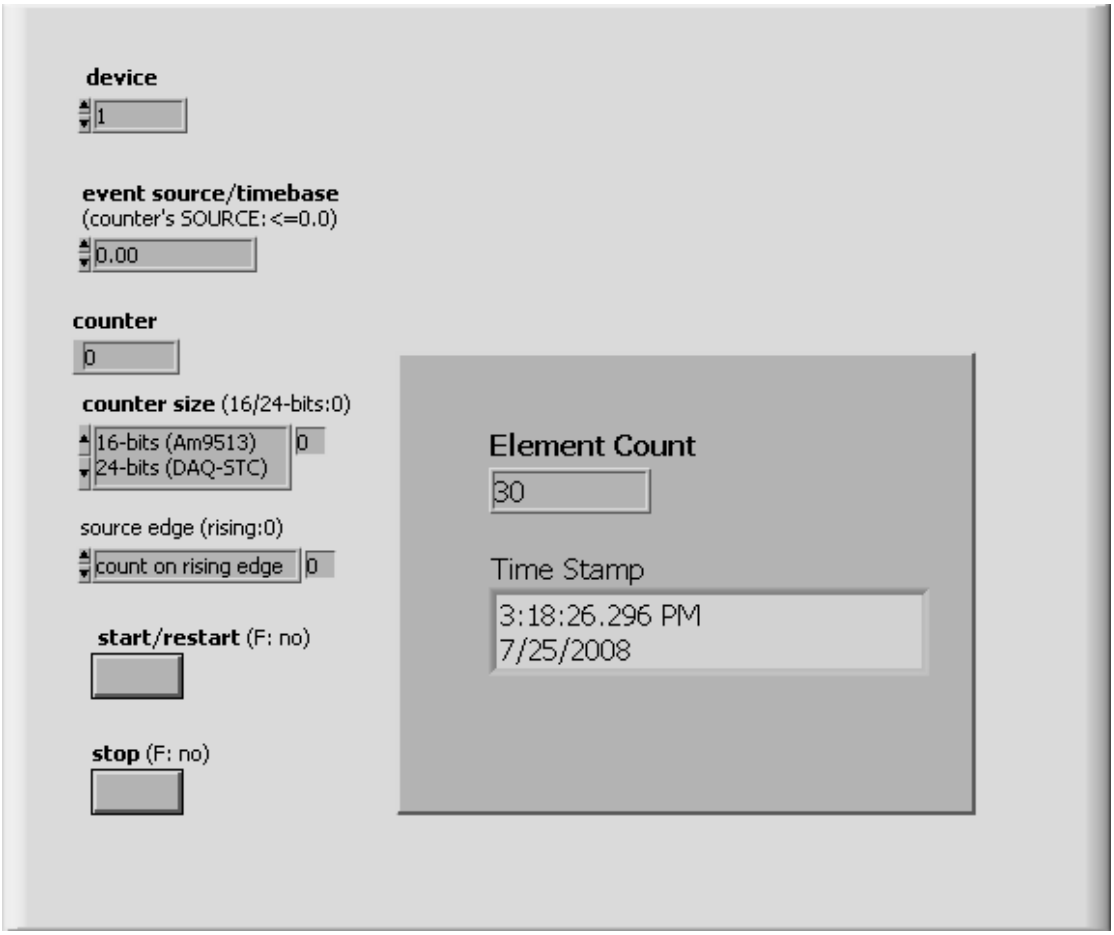


Figure D-1. Example Interface of Time Stamping

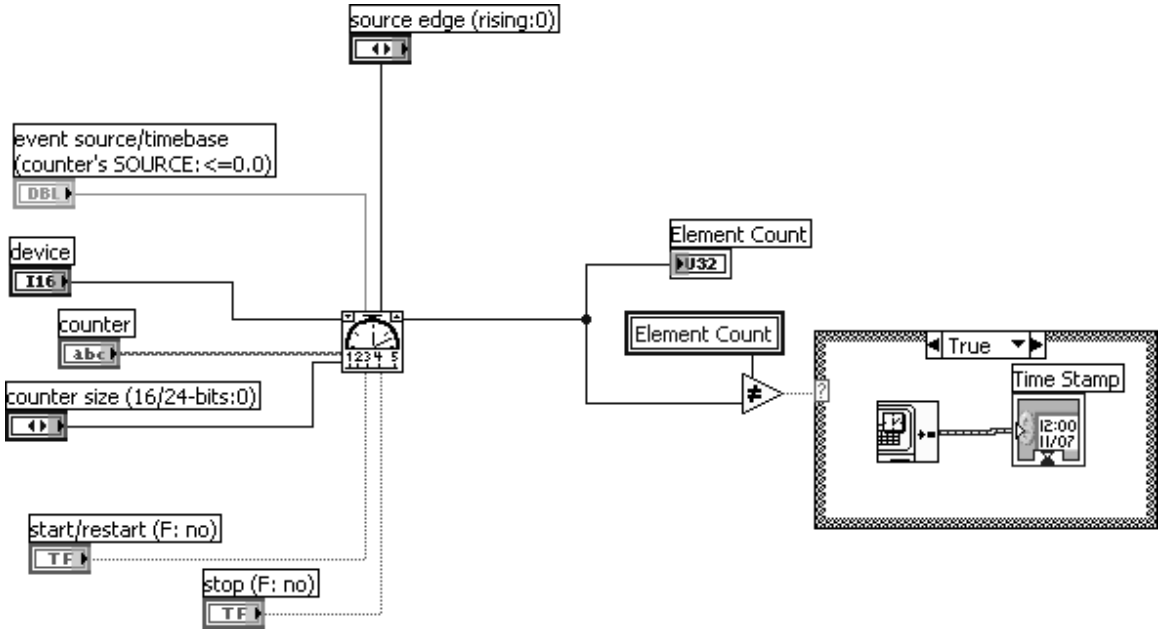
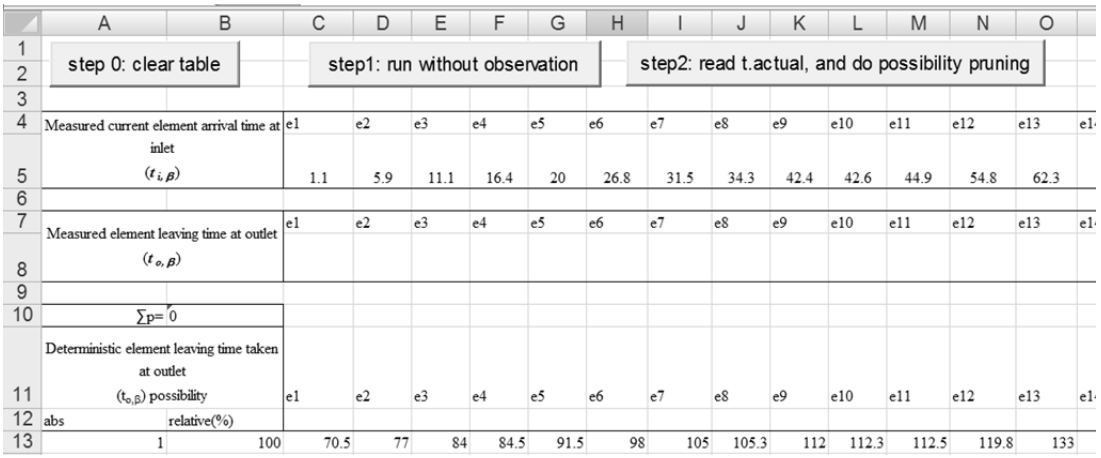
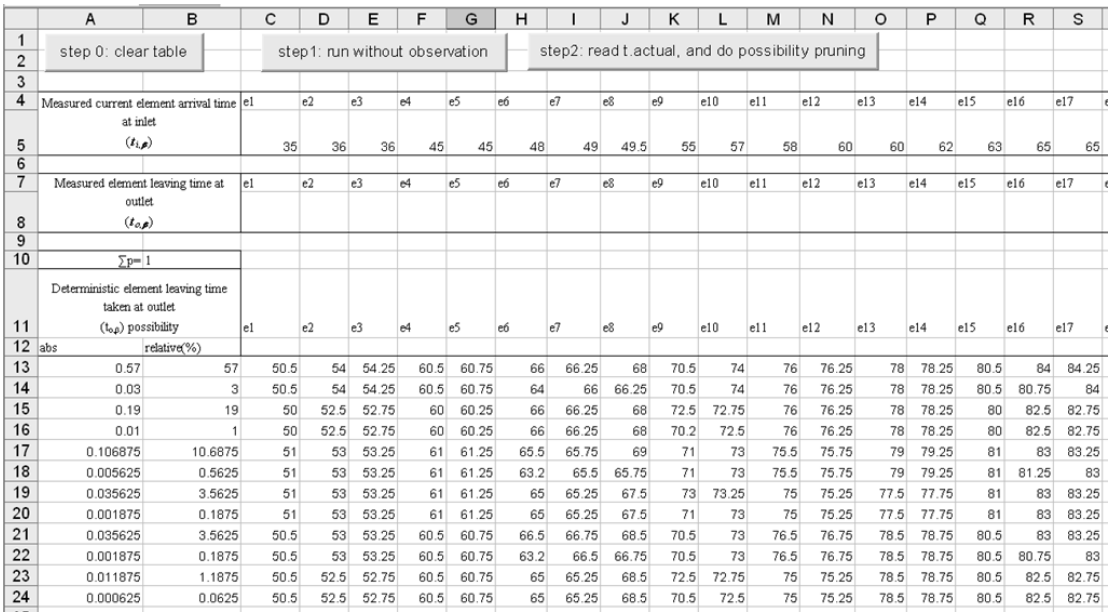


Figure D-2. Example Interface LabVIEW Program



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1															
2	step 0: clear table		step1: run without observation					step2: read t.actual, and do possibility pruning							
3															
4	Measured current element arrival time at inlet		e1	e2	e3	e4	e5	e6	e7	e8	e9	e10	e11	e12	e13
5	$(t_{i,\beta})$		1.1	5.9	11.1	16.4	20	26.8	31.5	34.3	42.4	42.6	44.9	54.8	62.3
6															
7	Measured element leaving time at outlet		e1	e2	e3	e4	e5	e6	e7	e8	e9	e10	e11	e12	e13
8	$(t_{o,\beta})$														
9															
10	∑p= 0														
11	Deterministic element leaving time taken at outlet ($t_{o,\beta}$) possibility		e1	e2	e3	e4	e5	e6	e7	e8	e9	e10	e11	e12	e13
12	abs	relative(%)													
13	1	100	70.5	77	84	84.5	91.5	98	105	105.3	112	112.3	112.5	119.8	133

Figure D-3. Example of Element Arrival and Leaving Computation



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1																			
2	step 0: clear table		step1: run without observation					step2: read t.actual, and do possibility pruning											
3																			
4	Measured current element arrival time at inlet		e1	e2	e3	e4	e5	e6	e7	e8	e9	e10	e11	e12	e13	e14	e15	e16	e17
5	$(t_{i,\beta})$		35	36	36	45	45	48	49	49.5	55	57	58	60	60	62	63	65	65
6																			
7	Measured element leaving time at outlet		e1	e2	e3	e4	e5	e6	e7	e8	e9	e10	e11	e12	e13	e14	e15	e16	e17
8	$(t_{o,\beta})$																		
9																			
10	∑p= 1																		
11	Deterministic element leaving time taken at outlet ($t_{o,\beta}$) possibility		e1	e2	e3	e4	e5	e6	e7	e8	e9	e10	e11	e12	e13	e14	e15	e16	e17
12	abs	relative(%)																	
13	0.57	57	50.5	54	54.25	60.5	60.75	66	66.25	68	70.5	74	76	76.25	78	78.25	80.5	84	84.25
14	0.03	3	50.5	54	54.25	60.5	60.75	64	66	66.25	70.5	74	76	76.25	78	78.25	80.5	80.75	84
15	0.19	19	50	52.5	52.75	60	60.25	66	66.25	68	72.5	72.75	76	76.25	78	78.25	80	82.5	82.75
16	0.01	1	50	52.5	52.75	60	60.25	66	66.25	68	70.2	72.5	76	76.25	78	78.25	80	82.5	82.75
17	0.106875	10.6875	51	53	53.25	61	61.25	65.5	65.75	69	71	73	75.5	75.75	79	79.25	81	83	83.25
18	0.005625	0.5625	51	53	53.25	61	61.25	63.2	65.5	65.75	71	73	75.5	75.75	79	79.25	81	81.25	83
19	0.035625	3.5625	51	53	53.25	61	61.25	65	65.25	67.5	73	73.25	75	75.25	77.5	77.75	81	83	83.25
20	0.001875	0.1875	51	53	53.25	61	61.25	65	65.25	67.5	71	73	75	75.25	77.5	77.75	81	83	83.25
21	0.035625	3.5625	50.5	53	53.25	60.5	60.75	66.5	66.75	68.5	70.5	73	76.5	76.75	78.5	78.75	80.5	83	83.25
22	0.001875	0.1875	50.5	53	53.25	60.5	60.75	63.2	66.5	66.75	70.5	73	76.5	76.75	78.5	78.75	80.5	80.75	83
23	0.011875	1.1875	50.5	52.5	52.75	60.5	60.75	65	65.25	68.5	72.5	72.75	75	75.25	78.5	78.75	80.5	82.5	82.75
24	0.000625	0.0625	50.5	52.5	52.75	60.5	60.75	65	65.25	68.5	70.5	72.5	75	75.25	78.5	78.75	80.5	82.5	82.75

Figure D-4. Example of Bus Catching Analysis Result