# An end-to-end data transformation process for increasing the information yield of system traces

# An end-to-end data transformation process for increasing the information yield of system traces

# An end-to-end data transformation process for increasing the information yield of system traces

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
rector magnificus, prof.dr.ir. C.J. van Duijn, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen
op maandag 3 juni 2013 om 16.00 uur

door

Kostas Kevrekidis

geboren te Kavala, Griekenland

Dit proefschrift is goedgekeurd door de promotoren:

prof.dr. M.J.Newby
en
prof.dr.ir. A.C. Brombacher

This thesis is number D169 of the thesis series of the Beta Research School for Operations Management and Logistics

A catalogue record is available from the Eindhoven University of Technology Library

ISBN: 978-90-386-9000-1

Printed by University Printing Office, Eindhoven

# Acknowledgements

Here I would like to take the opportunity to express my appreciation to the people who supported my work on this thesis.

I would like to thank my first promoter prof. Martin Newby for the many discussions we had on research problems of technical nature.

I would like to thank my second promoter prof. Aarnout Brombacher for the discussions we had on the story line and the readability of the thesis.

I would like to express my gratitude to the members of the PhD committee, prof. Frank Coolen, prof. Rob Kusters and prof. Geert-Jan van Houtum for their constructive feedback. In particular I would like to thank prof. Frank Coolen for providing me with detailed comments, which helped me to improve some technical aspects of the thesis. I would like to thank dr. Lu Yuan for joining the committee and prof. Will Bertrant for chairing the committee.

I would also like to thank the people who have facilitated this research. Firstly I want to thank dr. Guillaume Stollman from Philips Healthcare, for giving me access to data and facilities. Without his help the exploratory part of my research would not have been possible. I want to thank dr. Peter Sonnemans for his support and guidance at the beginning of my research and prof. Ton de Kok for allowing me to access the University's resources for an extended period of time. I am thankful for all the administrative help I received by Hanneke Driessen and the administration of the research group OPAC.

I heartily thank all my friends and colleagues who helped me with our small or long discussions on various research topics. I thank Kurtulus, Arvind, Aylin, Maurits, Christelle, Ilse, Joël and Jeroen. Kurt, I thank you for the great (intense) time we had writing that business proposal that was based on this research. We gave it a good shot. I also want to thank you for our discussions on some of my research problems.

Dad, Mom, Maria, thank you for supporting me non-stop on my entire journey.

Last but certainly not least, Karin, I have told you many times already, and I will keep on telling you, without your support this thesis would have been an equation with no solution. Thank you.

## Summary

Computer-based products are capable of recording and sharing information on the state of a product and its components, while the product is in operation. That information is acquired in the form of *traces.* Traces are a valuable source of information for identifying the causes of failures in computer-based products. Professional systems are such products, characterized by their large size and high complexity. There is a strong requirement for these systems to remain functional for long periods of time. These products are being developed and maintained with certain availability targets in mind. Information on the failures that these systems experience in the field can help to manage the resources effectively and meet the availability targets.

The traces generated by professional systems that are operating in the field, here referred to as *system traces*, have the potential of becoming an important source of information to support effective availability management. On operational level, traces can guide corrective maintenance activities by providing information on the root cause of failures. On planning level, traces can help to identify the main causes of system unavailability, which can lead to improvements in the system design or the preventive maintenance plan. In order to use system traces for this purpose, it is necessary to identify which of the many records in a long sequence of traces represent a distinctive physical event of interest, e.g. system failure.

However, because of the multiplicity of the components in these systems, as well as the long operating times, the amount of traces that is produced is beyond the capacity of human processing. In this thesis, a system trace *transformation* methodology is proposed that enables the systematic reduction of the data size of system traces without losing the relevant information that is useful for effective availability management. The data size reduction is achieved in two steps:

1- Clustering of raw traces into representations of single physical event instances;
2- Clustering of the latter representations into new representations of instances of physical event *types.*

The proposed methodology is generic and does not make use of system type specific information to drive the data size reduction. The methodology relies on unsupervised data mining techniques that operate on the features of the data structures found in system traces. The performance of the methodology is measured by the compression ratio, i.e. the ratio of the number of data points needed to convey the information after compression over the number of data points needed to convey the same information before the compression.

The research is organized in three main sections. In the first section the features of the data structures in system traces are explored. Three different approaches, namely domain experts, experimentation and graphical analysis, are used to increase the knowledge on the feature characteristics of the data structures found in system traces. The second section is utilizing the knowledge on the feature characteristics and proposes a set of methods and tools to systematically reduce the data size by organizing the data structures in the traces appropriately. The proposed methods have been developed to tolerate variation in the feature characteristics of data structures

when applied in the field. The third section is a case study. The proposed methodology is applied to a sample of system traces collected by X-ray scanners that are operating in the field. The case study is used to demonstrate the data compression that is achieved by applying the methodology on raw system traces and to assess the methodology's reliability. In the case study it is shown that the data compression ratio can exceed the order of 0.01, i.e. 100 times fewer data points are needed to convey the same amount of information as in the original sequence.

# Table of Contents

# List of Abbreviations

| | |
|---|---|
| AFP | Average number of False Positives |
| ASR | Average Success Rate |
| B2B | Business to Business |
| CP | Collision Probability |
| CSM | Cluster Separation Measure |
| DBST | Distance Between Successive Traces |
| DSR | Distortion to Signal Ratio |
| hCSM | Hybrid Cluster Separation Measure |
| ic | Identification Code (for traces) |
| KDD | Knowledge Discovery in Databases |
| LCC | Life Cycle Costs |
| LED | Levensthein Edit Distance |
| maxDBST | Maximum Distance Between Successive Traces |
| mDBST | Mean Distance Between Successive Traces |
| NED | Normalized Edit Distance |
| OEM | Original Equipment Manufacturer |
| PLC | Product Life Cycle |
| PPS | Partially Periodic Sequence |
| TCO | Total Cost of Ownership |

# Chapter 1

# 1 Introduction

Advanced professional systems are products that are used in the core processes of businesses. Examples of such products are medical systems, manufacturing or assembly machinery, baggage handling systems in airports, professional printers, etc. These systems are *capital producing*, they play an integral role in the productivity of businesses by affecting the yield of consumer goods or services. The interruption of normal operation of these systems, planned or unplanned, can have serious consequences not only for the health of the businesses but also for wider parts of society.

Recently an unexpected shutdown of a Canadian nuclear reactor (Chalk River, Ontario) that coincided with the scheduled maintenance of the Petten nuclear reactor in the Netherlands caused a worldwide shortage of medical isotopes [Reu10]. In a different example, system fault of the baggage system of a new terminal in Heathrow airport caused thousands of bags being stranded and flights being cancelled and delayed [Air08] [Tel08].

Costs accompanying such interruptions derive mainly from the loss of productivity, penalties imposed by breaking contracted deliverables and the cost of repair and spare parts. Original equipment manufacturers (OEMs) that produce these systems are aware of how important it is for their products to deliver their functionality uninterrupted. Therefore *availability* is perceived as a key performance indicator of these products. To provide products with high availability two attributes of the product's performance are considered:

- The frequency of system failures
- The period of time the system is unavailable as a result of a failure

To increase availability, one or both of these attributes has to be reduced. Reduction of system failures is achieved during design by using highly reliable components and by redundancy of critical components. Both tactics are considered during the design phase of the system. However, even when both design tactics are considered, 100% system reliability is practically impossible to achieve. Failures occurring during operation cannot be entirely avoided. These failures have to be handled by maintenance operations taking place in the field.

In the event of unexpected failure corrective maintenance is required. Corrective maintenance can be a time consuming activity because it involves failure diagnosis, spare part availability and logistics. To overcome the effects of unexpected downtimes, OEMs and businesses have turned to preventive maintenance policies. Under these policies, systems and their components are maintained on a regular basis before any failure occurs. This proactive approach has the advantage that the maintenance activity can be planned and prepared to reduce downtime to the minimum required.

In both cases, corrective or preventive maintenance activities are a result of the awareness about the system's state. Awareness of the system's state is created with the help of some form of signaling function. For corrective maintenance, that awareness can come for example from an audio alarm going off if the system malfunctions, or simply from the operator who detects the malfunction. For preventive maintenance, the awareness is made possible with the use of system monitoring techniques. These techniques allow the monitoring of the state of the system and its components. For example, information on the state of a component combined with engineering knowledge can help anticipate the component's failure. This technique is known as condition based maintenance. If a failure is anticipated, the component is replaced or repaired before that failure occurs and thereby system failure is avoided. For both types of maintenance, the signaling of the components or system's state is used to decide what action to take to reduce system unavailability.

For professional systems it is beneficial to avoid downtime due to system failure or maintenance altogether. In systems where much of the functionality is provided by software, engineers have implemented a protective tactic against failures, known as *system resilience.* System resilience describes the system's ability to either mitigate the effects of errors and confine the failure to retain as much as possible of the critical functionality, or to enable the system to recover from a failure as soon as possible back to its normal operating state as it was before the failure occurred. Whether the mechanism is error mitigation or failure recovery, engineers need to employ this solution where the system needs it most. Information on system failures can help engineers identify the functional areas of the system that need to become resilient.

Whether it is to guide maintenance activities, to improve the components reliability, or to design resilience into the system, information on the failures of the system is valuable. Such information can be obtained directly from the system. Computer-based systems are capable of recording and sharing information on the state of the system and its components, while the system is in operation. That information is acquired in the form of *traces. Traces* are a human readable, text based data form that contains semantic information on component level. Traces can contain information about the operational status of components or processes. Traces also contain the time when the recording took place. They are produced during operating time and their digital format allows them to be shared via network connection in almost real time.

Traces can also contain information about component failures and the recoveries that took place. For engineers this is a valuable source of information for understanding the system's behavior in relation to failures and recoveries. However, the extraction of such information from traces is not a trivial task. Due to the multiplicity of components and the running system processes, traces can contain a vast amount of data (thousands of entries) for a few hours of operating time. To manually inspect raw traces and identify instances of failures or recoveries is a time consuming exercise even when it is done for one system only. Sometimes it is necessary to inspect the traces from a fleet of systems to obtain a clear picture of a problem.

To make the amount of data manageable, the volume of raw traces has to be reduced to the point where only the most relevant data is retained. Moreover, the remaining data should be provided in a format that allows the direct application of some of the analytical techniques that are used in availability management. The whole process has

to be automated and made as generic as possible, free from product specific dependencies.

## 1.1 Motivation

This thesis is entitled “An end-to-end data transformation process for increasing the information yield of system traces”. The methods presented here seek to systematically reduce the size of traces using pattern recognition techniques, without losing any of the essential original semantic information and to transform them into a data format suitable for analytical post processing. The analytical methods and tools that are used for post processing are out of the scope of this thesis. Nevertheless, certain categories of such methods, namely stochastic modeling and sequential data mining, are considered and used as a reference for setting the requirements for the format of the data that should derive from the transformation of raw traces.

Since professional systems are in many cases managed in large numbers known as fleets, the combination of information from traces of multiple systems is being examined in the research.

The models and techniques are developed with the help of traces sampled from systems that are operating in the field. These sample set is used as a learning ground. Abstract data characteristics derived from the learning set are used for the development of a model that can represent basic characteristics of system traces. This data abstraction assures that our methods are not product specific but are based on system trace generic patterns. The developed models and methods are demonstrated and assessed using a case study.

The remainder of this chapter is organized as follows: first the concept of a system will be described. Then the importance of availability for professional systems is underlined in 1.3. In 1.4, the *system engineering* approach that is used to develop professional systems is described. The discussion includes how system availability is addressed during system design, and how fault resilience, and monitoring techniques are incorporated into that design. Also, in the same section, the role of the system environment in supporting the availability of the system is addressed and it is briefly described, how component state monitoring techniques are used to support the maintenance activities as well as the allocation of resilience. In section 1.6 current works on the area on the use of traces for system availability are discussed. Finally, in 1.6.2 we present the objectives of this research regarding the transformation of traces. More specifics about the objectives of the transformation process will be given in Chapter 2.

The research was carried out in cooperation with Philips Healthcare.

## 1.2 The System

The digital revolution brought by computer and information technology in the last thirty years had its impact on the design of capital goods. Most professional systems are now computer based, with data processing and storage capabilities. Software applications are to a great extent responsible for providing the system's functionality. Embedded software is controlling the operation of hardware components, and other software components are establishing the communication between hardware and software applications. Components are communicating with each other by passing on

digital information and signals. They provide the functionality efficiently by cooperating in a synchronized manner.

The system is modeled as a collection of physical components (electrical, mechanical, mechatronic, software) which are controlled and coordinated by software to achieve the system function. A module or a subsystem is a collection of components put together to provide a certain type of service. A basic assumption of this thesis is that complex systems follow a modular design, with strong coherence of components within modules and week coupling between modules. This assumption is generally true for complex systems. These systems are designed in a modular fashion to allow reduction of complexity during development and allow easier maintenance in the field. This assumption has to hold for the methodology in chapter 5 to be applicable. In this thesis the software and hardware are treated on an equal basis. Later in the thesis, it will be shown how the fault injection experiment (chapter 3) induces faults in software and hardware components to determine how they affect the system as a whole and how they affect the traces that the system is producing. Additionally, in this thesis the system availability is directly linked to component and process availability. The failure and unavailability of the latter directly implies failure and unavailability of the former. The recovery of the component directly implies recovery of the system. This assumption applies particularly to core functionalities, where the failure of a component makes the system unable to provide its service as required and therefore be considered unavailable.

## *1.3 Availability: key performance indicator for professional systems*

Availability is one of the most important performance aspects of professional systems because their operation is directly affecting business costs, throughput and quality. Availability is defined as the proportion of time that the system is operational. This section aims in underlying the importance of availability management of professional systems. The drawback of current data sources and the advantages of traces for supporting the decision making in the areas described in this section will be described in chapter 8.

### 1.3.1 Life Cycle cost and Total cost of ownership

The life cycle of the system is roughly divided into four phases in the following order: design & development, production, exploitation and disposal. Each phase has various types of costs associated with it. The cost of research and development are affecting the first phase and manufacturing costs the second. Utilization and maintenance costs are affecting the third and the costs of disposing the system come at the end of the life cycle. The cost throughout all phases is known as life cycle cost (LCC). More detail on the costs associated with each phase can be found in Blanchard et al. [Bla06].

The LCC that is paid by the owner of the system throughout its life cycle is known as total cost of ownership (TCO). Traditionally the customer is bearing the entire LCC. Design, development and manufacturing cost are included into the price of acquisition that constitutes part of TCO. Utilization, maintenance and disposal are a part of TCO too.

Studies have shown that 70% of TCO are due to downtime and maintenance costs, whereas downtime alone can account for 50% of TCO [Öne10] [Asi98]. Customers are becoming more and more aware of what effect system unavailability has on TCO, and are motivated to perform TCO analysis before they decide which system to purchase [Fer02].

### 1.3.2 From product acquisition to product leasing

The provision of service as a way of making business is becoming an increasingly interesting prospect for manufacturers. This model proposes customers paying for the functionality of the product and manufacturers being responsible for the continuous availability of that functionality [Mar05]. The model holds benefits for both customer and manufacturers:

- Manufacturers can create substantial revenue from managing their installed base. Services have greater margins than products and services provide a more stable source of revenue as they are resistant to the economic cycles that drive investment and equipment purchases [Oli03].
- Services are more difficult to imitate, they provide opportunities for differentiation and competitive advantage for manufacturers [Van88].
- Customers prefer acquiring the functionality than the physical product because this way they can reduce the TCO and their head count, and they can focus the attention on their core business. They also get access to technical system expertise and therefore can acquire improved service delivery and quality [Mar05].

Availability directly suggests functionality. For the above mentioned reasons OEMs are becoming very interested in providing to their customers systems with high availability.

### 1.3.3 High availability is perceived as good quality

The economic success of a product depends heavily on its quality. Eventually buyers will seek the brand with higher product quality [Mud02]. This is equally true for professional systems. Some of the aspects of customer perceived superiority of industrial products, which leads to product success is quality, reliability and availability [Coo79]. Moreover, product quality is a key element in winning customer loyalty. Product quality, perceived mainly as reliability and performance, is the main brand-equity-generating variable in the business to business (B2B) market [Ben04]. Product quality, reliability and of course availability are therefore major competitive attributes of the product.

Although availability is one of the main quality characteristics of professional systems, failures that relate to safety are for manufacturers more critical problems. In many cases manufacturers are legally bound to consider all possible safety issues that relate to the functionality of their systems. Solving a safety related problem would most certainly have a higher priority that improving availability. In this thesis the discussion is set on an availability centered basis; i.e. component failures that cause high system unavailability are ranked high in the list of problems to solve. Although in reality safety critical issues would top a list of availability related issues, in this thesis we treat safety critical issues as lack of availability.

## *1.4 Designing for availability*

Professional systems pose an engineering and economic challenge for OEMs. They need to fulfill customer needs by incorporating systematically, diverse technological knowledge into one economically sustainable and durable product solution.

### 1.4.1 The System Engineering approach

To engineer a professional system that fulfills customer requirements in a financially competitive manner, OEMs follow a system engineering approach [Bla06]. The approach takes requirements from all product life cycle (PLC) phases of the system into account and incorporates them in the design and development of the product (More detail about the PLC can be found in appendix A). Next to the system specific requirements, the system engineering approach also looks into the system support environment to identify components e.g. maintenance engineers, spare parts inventories, logistics, etc. on which the system's performance will depend. During the design and development phase the requirements of the system are considered together with the requirements of the support environment. The intention is to provide a system that can provide its functionality successfully during utilization, a support environment that can help sustain it, but also a system environment interaction that allows the best utilization of resources.

### 1.4.2 Availability requirements addressed by the system engineering approach

To fulfill system availability requirements, decisions regarding the frequency of system failures are addressed during the early stages of design and development. The reliability of components and the level of redundancy play a key role in the frequency of system failures. Critical components are designed with high reliability and two or more are put in parallel configuration to ensure the availability of the delivered function. The level of reliability designed into the components and the amount of redundancy in the system is based on the tradeoff between the cost of reliability and the required level of system reliability. Although these design tactics have been proved to be effective in improving system reliability, it is not unlikely that once a system has been introduced into the field, it is discovered that some components do not meet their reliability requirements and are the cause of frequent system failures. Such a discovery is backed by component failure information collected from fielded systems.

### 1.4.3 System resilience against failures

Recently, another design tactic has been added to the set of techniques for enhancing system availability, namely system resilience. The principle of resilience is that in the event of a failure the mechanism put in place will either confine the effects of a component's failure or restore the system to its prior to the failure state as soon as possible.

Resilience can be defined as "*...the intrinsic ability of a system to maintain or regain a dynamically stable state, which allows it to continue operations after a major mishap and/or in the presence of continuous stress*" [Hol06]. The concept emerged from the necessity and ability of organizations to anticipate exceptional events that can lead to accidents and mitigate their effect by taking appropriate action. This ability is particularly relevant for systems used in safety critical operations such as medical

intervention procedures on patients or air traffic control systems regulating the path of airplanes approaching and leaving the airport.

The key element of resilience engineering is the early detection of exceptional events, such as signs of imminent failure, in order to make necessary adjustments to the system as early as possible. The earlier the adjustment takes place the smaller the adjustments need to be.

Many professional systems have started to incorporate resilience engineering concepts into their design. When loss or degradation of the functionality is detected by an internal monitoring mechanism, the system initiates a "graceful degradation" process that limits the loss of its functionality. This way the system can continue to provide its core functionality without interruption of its operation, until the problem is addressed. Alternatively, if the failure cannot be contained and the system fails, internal *recovery* mechanisms can bring the system back to its functional state, by resuming the state of system prior to the failure. In software engineering the term is known as recoverability [ISO01]

For engineers it is important to know where in the system to implement resilience and how effective these mechanisms are in dealing with failures. Resilience engineering is benefiting from digital technology, which provides the means to monitor these mechanisms and assess their effectiveness.

### 1.4.4 System support environment preventive maintenance

The system support environment's role in system availability is decisive. In the event of system failure (and where no recovery is possible) corrective maintenance is required. The duration of system downtime depends on several factors such as the responsiveness of the maintenance engineer, the ability to recognize, diagnose and isolate the failure quickly, the availability and delivery time of spare parts etc. No different than the product quality, service quality in the B2B market has similar impact on customer satisfaction and affects repurchasing decisions [Pat09]. Service quality can depend on pre agreed deliverables. One of the dimensions of service quality is the ability of the service supplier to respond quickly to a customer's problem and to do things right the first time [Lap00]. Manufacturers of professional systems make decisions on how to configure the support environment to provide efficient levels of service. To reduce the chances of unexpected failure and to avoid the drawbacks of corrective maintenance's, preventive maintenance policies are adopted. During preventive maintenance components are replaced proactively to reduce the probability of failure or the degradation of their functionality. Although for hardware preventive maintenance comes in the form of replacement or repair, for software based components preventive maintenance can be perceived as correcting a newly discovered fault in the code and then uploading that fixed code version to all fielded systems before the failure actually occurs in any of those systems. The support environment is in many cases responsible for deciding which preventive maintenance policies can yield the required levels of availability.

## 1.5 Monitoring the state of components with the help of traces

To support the system design decisions and maintenance activities it is important to have information about the system operation on component level. Knowing where an error occurred, which components were affected, whether the system was able to recover or not, and how long the system was unavailable can guide the decision making on when, where and how to act. Corrective maintenance can be triggered by the signal of a failing component. The diagnosis that the field engineer has to perform can be guided by the semantic information that is collected by the monitoring mechanism. For preventive maintenance, information taken out from frequencies of component failures or condition monitoring techniques is paramount. An eminent failure can be anticipated by using the knowledge on how the component's state changes over time.

To gain such information engineers rely on monitoring techniques. Next to direct visual inspection, engineering methods to monitor the level of vibrations, the pressure, the noise or temperature, or the chemical analysis of fluids, can help to gain insight in a component's state. For computer based systems these traditional condition based monitoring techniques are not applicable. The state of digital components and software cannot be monitored by external observation. Instead, the software of the system is capable of reporting events occurring in the components. These events are recorded in an event log, known as traces. Traces contain semantic information on component states, e.g. errors or recoveries. This information is recorded together with a timestamp containing the time of occurrence of the event.

The monitoring and recording mechanism is embedded into the system's design. During design, “checkers” or “hooks” are put in place in the product’s software. Numerous checkers in the code target the behavior of a predetermined set of components. These sensing points are producing traces that contain information on the state of the components during operation. Traces are stored in the system and can be shared via a remote network connection. Sensing points can be added if that is deemed necessary to increase the capability to monitor a wider area of the system. However wider coverage comes with the cost of increasing the amount of data recorded. The information that is recorded in the traces can be used by engineers for post analysis to determine the background of certain events e.g. system failures or to help them understand the overall system performance.

The analysis of traces by humans is labor intensive. The ability of the system to record traces rapidly, the large number of components in the system and the numerous hooks in the software result to long sequences of traces that can contain thousands of entries for few hours of operation. For humans to use traces to monitor the operation of systems, these sequences have to be reduced to a manageable size that contains only the information of interest.

### 1.5.1 Analytical methods for assessing system availability

The aim of this thesis is to prepare raw traces for analysis that seeks to improve the availability of the system. Firstly traces are used for failure diagnosis after a system failure. This use of traces is fundamental. The ability to use the information in the traces for failure diagnosis has to be retained after their transformation. Beyond this

use, in this thesis two types of analytical methods are considered. Engineers can use these methods to determine if and where the system requires attention in order to meet its availability targets.

The first of these methods is stochastic availability modeling. In stochastic availability modeling, densities of failure and repair/recovery times are modeled and used to analyze design, performance or maintenance scenarios. The result of the analysis can help to decide on which design is superior or which maintenance policy will yield higher availability for lower cost. For this type of analysis traces can provide the empirical input for determining the type of probabilistic failure models that are suitable for system modeling and for the parameterization of these models. Traces can be also used to compare actual against expected performance.

The second technique is known as discovery of association rules and is a type of sequential data mining. Association rule discovery can give answers to questions such as "which recovery processes are associated with which types of failures". This information can help engineers assess whether the system resources are allocated effectively to deal with errors and prevent system failures.

The application of these techniques in conjunction with the use of system traces are described in more detail in Chapter 8.

To support the two techniques, failure and recovery events have to be provided in the form of point representations. In raw traces, the instances of the physical events, failures or recoveries, can be represented by multiple traces. The collection of one or more traces resulting from the occurrence of a failure or recovery is knows as *subsequence* (the term will be defined in Chapter 2). Subsequences can have nonzero duration. Given a physical event, there length of time between the moment the first relevant trace is recorded till the last trace related to the same event is recorded can be greater the zero. To meet the requirements of the above mentioned analytical post processing techniques, subsequences have to be represented by points with a single temporal location and of zero duration.

## 1.6 The role of traces in system availability related research

Availability analysis of complex systems using traces has been a topic of interest for researchers from the field of computer system engineering and network management [Sim05][Mar05][Mor90][Kal99][Tal99]. The challenge in extracting the relevant information from long sequences of traces is to identify, without manual inspection which traces form the subsequences that represent the physical events of interest, i.e. error and recoveries.

Different techniques have been used for identifying instances of physical failures in long sequences of traces. In some occasions physical failures are represented by a single trace. In [Sim05] measurements of product availability are retrieved from long sequences of traces by using the time of occurrence of single traces, which in turn can represent the occurrence of a physical error and a crash. A mapping of the semantics onto failure types was conducted in advance, which allows the identification of the relevant single traces in long sequences. A similar approach is being followed by [Mor90]. However, there the system recoveries are not recorded. The availability is estimated assuming a fixed duration of downtime for each occurrence of a failure. In

[Kal99], availability is estimated using traces, with emphasis on the recovery recordings. In this study too, categorization of trace precedes the quantitative analysis. In the above studies physical failure events or crashes are identified in long sequences of traces because the semantics of the traces of interest have been either linked to failure types in advance or other product specific knowledge has been added in the analysis. The availability of a software system is assessed in [Mur95] by using the time between system crashes using traces. However, traces that contain information on the cause of the crashes i.e. failures are not taken into account in the analysis

In another line of studies, the semantics of subsequences that represent errors have not been linked to external knowledge on failure or recovery types. However, other information about the traces is known. Cinque et al [Cin05] are simulating user profiles and produce traces on occurrences of failures. Multiple traces are clustered into one group to represent one physical failure. The clustering is performed based on the knowledge about the starting and ending point of the subsequence of traces. In [Ham03], single error traces are classified into categories of failure types by searching for key words in the content of the error message. Then the types are correlated to system states. A similar approach is used to analyze outages in a university network by [Cho07] and to analyze the availability of processors [Qua00]. Clustering methods for traces without the use of any prior knowledge on the content of subsequences are described by Tsao [Tsa83] and the extensions made by Hansen [Han88]. The clustering is based on the temporal distances between successive traces found in long sequences. This approach is used as basis for the methodology that is proposed in Chapter 4.

Beyond availability measurement or analysis, traces have been used in other areas, such as root cause analysis and fault diagnosis [And95] [Laz92], failure detection and prediction [Lim08][Tha96][Tie00], analysis of system and network of systems behavior by correlating events representing alarms [Bel] [Yam05], and model validation [Wei90]. These latter studies try to correlate a single trace to a physical error. They are depending heavily on prior knowledge on the description of failure symptoms.

### 1.6.1 Current methods

The manual extraction of information from traces is labor intensive, making the processing of traces from large scale applications virtually impossible. Even for the processing of traces from a single system an engineer would have to search through thousands of log entries to find the most relevant and decide how to organize them in a meaningful for the purpose manner. Such a manual operation is meaningful if information from traces is needed to help the diagnosis of a single system failure event, where a short strip of the traces has to be analyzed. However when the required information is about the occurrence of multiple failure events, the process of manually analyzing traces even for one single system goes beyond the capabilities of humans.

For this reason automated methods are developed to process traces and extract the required information. The methodologies proposed in literature are mostly ad hoc solutions to specific problems (exception is [Tsa83]). They depend on case specific knowledge to decode the information in traces, i.e. a priori mapping of trace onto failure types, key words in the description field, start and stop marks etc. Such

knowledge can be collected with the help of exploratory exercises. For example, extensive fault injections under specific operating scenarios can provide a collection of error and recovery traces. These can be used for identifying the same fault in the trace recorded in the field. But such an approach can produce only a finite set of fault traces. Covering all possible faults scenarios under different operational conditions is practically infeasible due to the high number of components and their complex interdependencies. The second way of obtaining additional knowledge is when the system is operating in the field. Engineers can perform root-cause analysis to relate traces to the physical fault events. Root cause analysis requires dedicated resources to investigate traces.

The above techniques are putting heavy requirements on resources especially because the investigation of traces is not directed to the most prominent failures, but follows a gunshot approach i.e. investigate traces as they occur without prioritization given the criticality of the event. A more efficient approach is to focus the investigation on a narrow group of traces that seem to be most relevant to system unavailability. For that the failure events with high frequency of occurrence or/and the events that cause long down times can be singled out using the temporal information in traces. Then, the semantic information can be used to guide root cause analysis.

Many studies that are using traces are focusing on either error or recovery traces. The association between the two types of traces is not examined. However, for supporting design and maintenance decision making, both types of traces are relevant. The information retrieved from traces can guide engineers to choose between changing the design, i.e. component reliability, resilience or enhance the maintenance of the system i.e. guide corrective or preventive maintenance. Therefore both types of traces should be included in the transformation methodology.

## 1.6.2 Proposed approach

The objective is to develop a methodology for reducing the size of raw traces through a series of transformation steps so that human interpretation of the data is enhanced. Other than the requirements that are described in the sections above, the methodology will have to take into account some other considerations:

### 1.6.2.1 Black box approach

A black box approach is followed in the methodology of this thesis. System specific information is not used as an input for developing the methodology. In the literature the black box approach is referred to as measurement based (see below), whereas the approach that is using design details is known as model based.

Model based approaches rely on analytic or simulated models of the system or its behavior. This approach requires design information that describes components and their interactions. Once defined, a model based approach is useful in exploring the system behavior under different scenarios. However, the accuracy of the results depends on how realistically the model represents the system and its behavior. Given the increasing complexity of systems, accurate modeling has become a difficult task [Mar05][Tri08]. Moreover, model based approaches still depend on measurements for estimating their parameters and for validation.

A measurement based approach relies on observations made directly on the traces. It has the advantage that knowledge can be acquired from the data collected from the field without having to know the system in great detail. The approach relies more on the understanding of the data, their structure and the information they carry. Such an approach can provide accurate results and is particularly useful when it is applied to operational systems [Iye00]. In this thesis observations and measurements are made on two basic characteristics of traces:

1. The temporal distance between successive traces (1.6.2.1.1)

2. The association between pairs of traces (1.6.2.1.2)

To establish our understanding on these two characteristics, an extensive exploratory study is conducted using traces collected from systems operating in the field (chapter 3).

#### *1.6.2.1.1 The temporal distance between successive traces*

Traces contain the timestamp of the occurrence of the event or state they represent. Multiple traces form an ordered sequence of events. Their temporal order and the temporal distance between traces define the temporal structure of the sequence. A sequence can consist of traces that are put densely next to each other, e.g. a few seconds between two successive records, or can have long intervals between two successive records, e.g. several seconds between two successive records. Typically when the system is active the distance between successive traces is short i.e. traces are recorded rapidly, whereas when the system is idle the distance between successive traces tends to be long. The temporal characteristics of the formations found in long sequences of traces are one of the main features that can help develop a generic, system independent methodology for identifying failure and recovery events. The temporal structure of traces is explored in chapter 3 and used to develop the segmentation method presented in chapter 4. The method is enhanced to deal with the variation in temporal structures found in real life applications.

#### *1.6.2.1.2 Association between pair of traces*

Professional systems can be highly complex. The complexity of the system derives from the number of components, the diversity of technological fields from which these components originate and the interactions between these components. To manage design complexity, professional systems use modular system architecture. The modularity allows the management of system complexity by organizing components into modules with inherent strong functional dependency. The modular system design is characterized by the principle of *strong coherence* among components that are part of the module and *weak coupling* between components of different modules. The *strong coherence/weak coupling* principle has a direct effect on the formation of subsequences of traces. Failure and recovery events usually involve components of the same module, due to the effect of failure propagation or recovery protocols. Therefore, the traces produced by failure and recovery events can represent these functional dependencies. Subsequences are formations of traces that can be attributed to one event. The subsequences of traces that are produced by failure and recovery events are manifestations of the *strong coherence/weak coupling* system design principle. Given that the system design remains fundamentally unchanged over its lifecycle, the *strong coherence/weak coupling principle* can be considered as a

constant reference for the formation of subsequences. This reference has model like properties; i.e. given a system design and assuming the same initial system state, every occurrence of the same failure or recovery event would produce the same subsequence of traces. The proposed methodology uses the generally applicable *strong coherence/weak coupling principle* in system design to develop an approach for processing long sequences of traces. This principle is explored in chapter 3 and applied in the methodology of chapter 5.

#### 1.6.2.2 Variations in temporal structure and associations

Professional systems do not operate in isolation. Professional systems interact with their environment: operators, other systems and external devices. The interaction with the environment together with system specific conditions can make the structure of subsequences vary for different instances of the same physical event. Similar variation can be seen in the associations between traces. This observation is made in chapter 3 and is taken into account for developing the processing methods discussed in chapters 4 and 5.

#### 1.6.2.3 Sequential processing

Among the methods used in applications that operate on traces, two distinctions can be made: the online and the offline set up. Online methods apply to incoming traces, in close to real time manner. Off-line approaches work with a snapshot of the traces over a period of time taken from databases. The decision to use an online or offline method depends on the latency requirements for system state awareness and the rate with which the system state can change. In the context of availability management, a real time approach addresses the needs of maintenance operations where the awareness on the system state can help corrective maintenance in fault diagnosis. When management control and long term planning are the dominant objectives the time horizons are longer and therefore information can be delivered in longer time intervals [Kee78]. Because the methodology should provide support for both design (long term) and maintenance, and it has to satisfy the stricter requirements of the latter and therefore be able to process data in close to real time manner.

The proposed methodology applies in the following manner: parameter estimation and algorithm calibration is performed off-line using snapshots of system traces. Application of the methodology on operating systems is online and close to real-time. The two modes are discussed in more detail in chapter 8. The latter objective requires the processing methods for traces to follow a sequential model; i.e. data are processed as they arrive into the data repository. The sequential nature of the process is shown in chapter 4.

#### 1.6.2.4 Combining traces from distributed systems

The proposed methodology is most likely to be applied to fleets of professional systems. That means that data from traces of multiple systems can be combined. For example, the training of the algorithms can be based on data collected from various systems. Combining data would allow the faster training of the mining algorithms, because more observations would become available in shorter time.

For that purpose, it is important that the data from various systems do not contradict each other but rather work complementary. However, environmental factors can have an effect on the formation of subsequences. Even though the system design is a

constant parameter across the distributed systems, the effect of the environment can compromise the suitability of a combinatory approach.

In this thesis a method to assess the consistency in the structure of subsequences obtained from distributed systems is proposed. This is method is presented in chapter 6.

## 1.7 Overview of objectives

An overview of the objectives for the methodology is given below:

1. The methodology seeks to reduce the size of the data via a series of transformation steps making it easier for human interpretation (chapters 3, 4, 5).

2. During the transformation all relevant information has to be retained. In the context of availability management that means that temporal locations of events have to be known after the transformation. Also the original semantics that relate to an event need to be available after the transformation so that they can be used for further investigation. The transformed data should represent the physical events as accurately as possible (chapters 3, 4).

3. The transformed data should be in a format that is suitable for post-processing by analytical methods. When the method is applied is should rely little on manual work. These objectives have to be reached under a set of conditions (chapter 4):

4. The methodology has to be generic, applicable to traces of any large systems (that have modular design) without the need to incorporate system specific information (follow a black box approach regarding the system). The methodology should rely only on generic characteristics of traces (chapter 4) together with generic system design principles (chapter 5).

5. The methodology has to be able to deal with variations found in the structure of the subsequences of traces. Time dependent factors such as the system load or type of operation can result to different trace manifestations of the same physical event in different instances.

6. The methodology has to be able to be applicable for close-to real time processing of traces (chapters 4 and 5). In the context of availability management, the most demanding requirement regarding the latency of information comes from the need to react quickly in the case of corrective maintenance. The transformed traces need to be available quickly to support this type of maintenance.

7. The methodology needs to exploit the fact that traces are collected from multiple systems of the same type operating in the field but are geographically distributed (chapter 6). The information that is collected from the traces of multiple systems can help to speed up the process of parameterizing the algorithms that are used for the transformation of traces.

## 1.8 Outline of the thesis

The thesis is outlined as follows: in chapter 2 traces are introduced formally. Their origin, their temporal and semantic structure is described. In the same chapter the *knowledge discovery framework* is introduced. The framework describes a high level process of how to make sense of large volumes of data, a problem that applies here too. This framework puts the basis for developing the particular methods and tools that will allow information extraction from traces. The methodology that derives from the framework consists of three steps: exploration of traces, sequence segmentation and subsequence matching. Chapters 3, 4, 5 are dealing with each one of these steps respectively. In chapter 6 the option of using traces from multiple systems as a method to increase the efficiency of the parameterization of the algorithms is considered. The option relies on whether the structures of subsequences across distributed systems have consistent characteristics. Chapter 7 contains a case study where the methods that are presented in chapters 4, 5 and 6 are applied to traces that have been collected from systems operating in the field. The case study serves also as a form of validation as it allows the evaluation of the results of the transformation and the performance of the methodology. Chapter 8 describes in more detail how the transformed traces can be used for post processing analysis. The transformed traces can help to create an entire decision support system for availability management. Such a system is described in chapter 8. Finally chapter 9 concludes the thesis with the findings of this research and a discussion on the proposed methodology.

# Chapter 2

# 2 Knowledge discovery framework for traces

To achieve the objectives stated in 1.7 a step by step process is defined that will perform the end to end transformation of raw traces. The sequence of the steps is designed around a framework that is borrowed from the domain of *knowledge discovery in large databases* (KDD).

This chapter is organized as follows: First, the basic terms used to describe traces, examples of traces and a brief description of the mechanism behind traces are presented in 2.1. In section 2.2 traces are generalized and described as *event based data sequences*. This view highlights the importance of the "evolving character of records" in traces, an informative aspect that is relevant for this research. The concept of variation found in traces is described in 2.3. In section 2.4 the framework to process the traces is presented. First in 2.4.1 the generic KDD framework for processing large data sets is presented. This generic framework puts the ground for defining a series of steps for transforming traces into a data format suitable for analysis. In section 2.4.3 the end–to-end transformation process for traces is defined. Finally the chapter closes with section 2.5 where the sample set of traces that is used in this research is described.

## *2.1 Traces*

A single trace contains information about the state of a component in the system at a certain point in time. This information is in the form of human readable text format (see example of trace in Table 2-1). Such a message describes briefly the state of the component, for example "processing data" or "data processing completed". The message can contain static and/or dynamic information. Static information consists of a predefined description that remains the same in different instances of the trace. For example "processing data" is a description that appears in that form in all instances whenever this trace appears. Dynamic information in the description can take the value of a variable, for the level of usage of a resource. For example, "memory usage at 90%" is a description that contains the variable of the percentage of usage. This value may differ in different instances of the message. In this research the dynamic information i.e. the variables are ignored and in relation to traces the term "description" is a synonymous to the term "message".

Traces have a distinctive identification code known as *trace identification code* or *ic*. The trace *ic* is uniquely linked to the description. For static information that means that the *ic* represents also the description field of the trace. The *ic* is also uniquely linked to each component. The trace *ic* helps to quickly identify the qualitative information of a trace without having to read the message. Trace *ic* are used extensively in this research as they allow fast machine processing.

An important element of the trace is its *timestamp*. The timestamp provides the information about the date and the time the trace was logged. Logging mechanisms can record traces with timestamps with a granularity of millisecond. This allows the rapid logging of traces and reduces the chance of having traces recorded with the exact same timestamp. However having traces with the exact same time stamp is common.

| timestamp | ic | message/description | class |
|---|---|---|---|
| 02-10-2007 9:27:31 | 730000001 | InfraToolsSaveDevData startup completed | Information |
| 02-10-2007 9:27:36 | 760000200 | Archiving:Communication Lost | Error |

**Table 2-1 Form of single trace**

Another element of interest is the field *class*. *Class* describes whether the trace represents an *informative* state of the component, i.e. a normal operation state of component, or an *erroneous* state, i.e. abnormal operation. Additional fields can describe the *mode* of the system, e.g. normal operation, start-up etc., the source code that produced the message and other. These fields are used for the initial filtering of the traces of interest e.g. the field *class* is used to separate the "Error" and "Recovery" traces from the rest. The field *mode* allows the exclusion of specific system modes such as start-up and shut-down, to retain only traces produced during normal operation.

Traces are recorded when hooks in the software are triggered. Triggering occurs when components enter certain states that have been predefined. The "sensing" of states is done either with the use of electronic, mechanical sensors or programming sensors [Tie00]. The latter are found in the firmware (embedded software) of components of professional systems [Len02] or the application layer.

When a sensor is triggered, the message that relates to the sensed component state, together with the values of the other fields and the trace *ic*, are sent to the logging unit that stores the information together with a timestamp. Several "sensors" are distributed throughout the system. As the system is operating, this network of sensors produces a chronologically ordered *long sequence* of traces (example of sequence in Table 2-2). Traces are also known as *system event logs*, and the process of sensing, reporting and recording is known as *system event logging* or simply *logging*. In this thesis the entire mechanism for logging is referred to as the *logging mechanism*.

| **timestamp** | **ic** | **message** | **class** |
|---|---|---|---|
| 19-06-2008 14:58:44 | 570000020 | Initialize job queue | Information |
| 19-06-2008 14:58:45 | 540019921 | Command: SelectRevExam | Information |
| 19-06-2008 14:58:46 | 650028673 | Monitor: Starting Application completed | Information |
| 19-06-2008 14:59:05 | 73200000 | ISB_FRONTAL | Information |
| 19-06-2008 15:03:50 | 73700000 | Frontal Channel | Information |
| 19-06-2008 15:03:53 | 73600000 | Frontal Channel | Information |
| 19-06-2008 15:05:04 | 510020389 | Cumulative dose values | Information |
| 19-06-2008 15:05:04 | 510021391 | System not ready | Information |
| 19-06-2008 15:05:05 | 510021393 | System ready | Information |
| 19-06-2008 15:05:06 | 540019921 | Command: CloseExam | Information |
| 19-06-2008 15:05:07 | 510028691 | Applied tube protection | Information |
| 19-06-2008 15:05:08 | 510028691 | Applied tube protection | Information |
| 19-06-2008 15:05:09 | 510019388 | Fluo flavour selection completed | Information |
| 19-06-2008 15:05:10 | 510999921 | Command: XTraVisionReadyStatus | Information |
| 19-06-2008 15:05:11 | 510999920 | User msg: Total free space 203410 (51316) images | Information |
| 19-06-2008 15:05:12 | 510021393 | System ready for fluoroscopy x-ray acquisition | Information |
| 19-06-2008 15:05:13 | 510021390 | System ready for exposure x-ray acquisition | Information |

**Table 2-2 Sequence of traces**

The connection between traces and physical events is not explicit. Traces do not contain information that relates them to distinctive physical events. This relation has to be established with diagnosis. In this thesis it is assumed that such connection exists i.e. physical events cause the logging mechanism to produce traces. When one or more traces are the result of one distinct physical event, they are considered to be forming a *subsequence*. A general example of a physical event is the *archiving of a file*, *a system crash*, or the *restarting of the system* after the crash. These are events that are technically interesting and that engineers use to describe the operation of the system. In Table 2-3 a subsequence is shown that contains traces describing the event of *archiving a file*. The physical event can have duration as it is the case in the example. The timestamps of the subsequence provide the information about the time of occurrence of the event e.g. the event started at 5:03:09. The semantics of the subsequence allow the understanding of the nature of the event. From the semantics of the subsequence in this example the connection between the subsequence and the physical event is clear. This is not always the case. Particularly error subsequences require further diagnosis into the problem before this relation is clear.

| timestamp | ic | message | class |
|---|---|---|---|
| 02-10-2009 5:03:09 | 670000001 | Open database connection | Information |
| 02-10-2009 5:03:10 | 570000021 | Execution of job started. | Information |
| 02-10-2009 5:03:11 | 570000015 | Archiving Job Completed. | Information |

**Table 2-3 Subsequence containing representing the archiving event**

The information in the traces is categorized into the *semantics* that refers to qualitative information such as the trace *ic* and the description, the class etc. and the *temporal* that refers to the date/time information found in the timestamps. The same terms are used to refer to derivatives of the above. For example “time between successive traces” is also temporal information.

## 2.2 Event based data sequences

Traces are a type of data form known as *event based data sequences.* The value of event based data sequences lies on the fact that they "enable the understanding of the evolving character of records in a data set" [Vro10]. The records, the traces, consist of the semantics (*ic*, or description) and the temporal information (timestamps). The semantic information provides the qualitative aspect of the event (what happened and where) and the temporal information provides the temporal aspect of the event (when did it happen). Put in a sequence, the semantic and temporal information of traces form the *evolving character of the records*. In the context of system operation, the evolving character of records represents the evolving states of the components in the system. Such records can be revealing and concealing the same time. They can be revealing by allowing a view into the states of components that *can* report on their state. Traces can be concealing, because components that cannot report on their state are not found in the records and they can be easily omitted.

To illustrate this, an example is provided. Figure 2-1 shows the evolution of the states of a system in 7 consecutive points in time. The system is represented by the blue panels. Each panel represents a point in time of the system state.

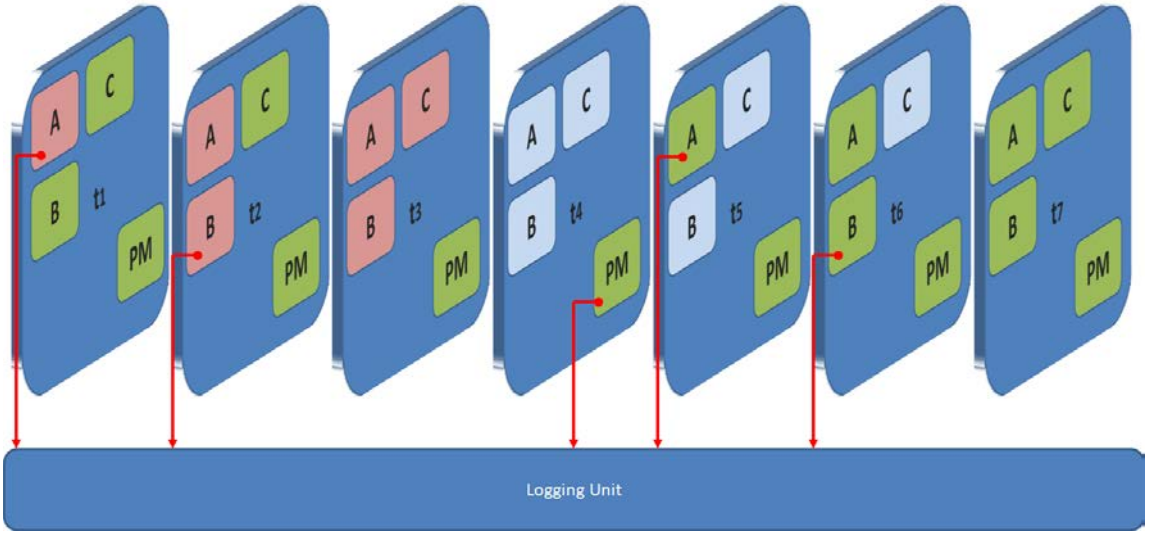



**Figure 2-1Evolving states of components**

The points in time are marked by the notation $t_i$ located in the center of each panel, where $i = 1,2,3,4,5,6,7$ . The system contains four components A, B, C and PM. Components A, B, C together provide the functionality of the system. Component PM is the process manager of the system, responsible for resource allocation and the synchronization between the components. Some of the components can report on their state and they can send this information to the logging unit. This operation is indicated by the red arrows that connect a component to the logging unit. The components with the logging ability are A, B, and PM.

At time $t_1$ component A is encountering an error and fails. Because of functional dependency, components B and C follow and fail at times $t_2$ and $t_3$ respectively. At time $t_4$ the PM detects that the system service is down, and orders components A, B and C to restart in order to restore the service. Components A, B and C, following a recovery protocol, restart progressively at times $t_5$ , $t_6$ and $t_7$ respectively.

The traces that are produced by the logging unit are of the form (the format is $< State\ Component, time >$):

$< Error\ A, t_1 >$

$< Error\ B, t_2 >$

$< Start\ Service\ PM, t_4 >$

$< Starting\ A, t_6 >$

$< Starting\ B, t_7 >$

The traces reveal the progression of states. The semantic information reveals which components were involved in the incident and what their states were (what happened and where). The temporal information in the traces reports on the latency of the events (when it happened). Overall it is understood that the system experienced a failure at

time $t_1$ and the system recovered at time $t_7$. From the above the evolving states of the system can be understood.

However, the sequence in the example does not contain any trace of component C. Component C cannot log traces and therefore its states remain unrecorded. If this component would be the cause of the system failure, the traces would not be able to help the diagnosis of the problem. It is a challenge to find the right balance between covering all interesting component states by increasing the instrumentation of the system with sensors and software hooks, and the same time keep the effort of instrumenting low and the amount of data manageable. The amount of data that are recorded depends on the logging mechanism that is responsible for the sensing, reporting and recording of component states. The more instrumentation the system contains, the more data there is to analyze and bigger need to automate their analysis. Usually systems have numerous of software hooks and the sequences that are recorded by one system can contain a great amount of traces. An hour long sequence of traces from one system can contain tens of thousands of entries.

Event based data sequences differ from another well-known computer based information source, the *core dump*. The core dump is a file that contains information on the state of the memory and/or the program at specific point in time e.g. a system crash. The core dump is produced only when such an event occurs. Traces on the other hand are incessant representations of the system's components during operation.

## 2.3 Variation in subsequences

The structure of subsequences i.e. the number of traces, the type of semantics the time between successive traces, is subject to variation. This variation results from the fact that under operational conditions the system can react, to a certain extent, in different ways to the same physical events. For example variation can occur due to the path an error propagates through the system. If the same fault occurs twice in the same component in two points in time, a different number of components might be affected in the two instances due variation in the error propagation. This difference can be motivated by the difference in the operational state the system at different instances. Similarly, in different instances of the same fault, the same number of components can experience errors in different sequence or with different time elapses between the component errors.

Additional variation can derive from the dynamics in the logging mechanism itself. As errors are sensed and the messages are sent to the logging unit, the logging can introduce variation in the sequence of logged traces or in the time of logging. Variation is found in the traces obtained from the same system and in traces of different systems. Variation in subsequences is taken into account when developing the transformation methodology. The variation in subsequences is explored in chapter 3 using experimentation. In chapter 4 and 5 it is discussed how the variation in subsequences can affect the results of the transformation. The proposed methodology incorporates features that enable it to account for the variation found in subsequences.

## 2.4 A sequential process for making sense of large amounts of data

"*Drinking from the fire hose of knowledge*" and "*data asphyxiation*" are phrases often used in literature to signify the wealth of information stored in databases and the difficulty of extracting useful information from it. Knowledge discovery in databases (KDD) is the field of research devoted to methods and tools that enable the efficient sense making from large amounts of data [Fay96]. Knowledge is discovered in the form of patterns in the data that can describe a phenomenon. An example is the consumer behavior patterns found in supermarket data and the consumer's tendency to buy certain products together. In event based sequences in particular, patterns are found in the timing or ordering of events. The discovery of such patterns is information that is directly usable, like the consumer behavior example, or it can lead to the discovery of usable information [Fra92]. The focus of KDD to discover patterns in large data sets is the attractive aspect of this framework that makes it suitable for the objectives of this research.

KDD has a discovery process that is made up from a series of steps that can lead from raw data sets to knowledge discovery. This thesis makes use of the KDD framework to develop a step by step transformation process for system traces. In the following section 2.4.1 the general KDD process is presented briefly, focusing on its most relevant aspects. In section 2.4.2 some elements of the KDD are discussed in detail. These elements were used strongly in this research. In the last section 2.4.3 the discovery process is described as defined for needs of this thesis.

### 2.4.1 The KDD discovery process

The knowledge discovery process consists of a series of steps that start from the raw data and end in the creation of knowledge. In Figure 2.2-2 the steps of a KDD are represented schematically. The raw data can be recordings of various types e.g. events (purchases, orders etc.), entities (income, professional skills etc.) that are stored in large amounts in databases or repositories. The data are structured in the sense that they have a specific format e.g. events are recorded in a particular calendar time format and there can be relationships between the records e.g. professional skills are linked to income. In each step of the KDD the data undergo a transformation.

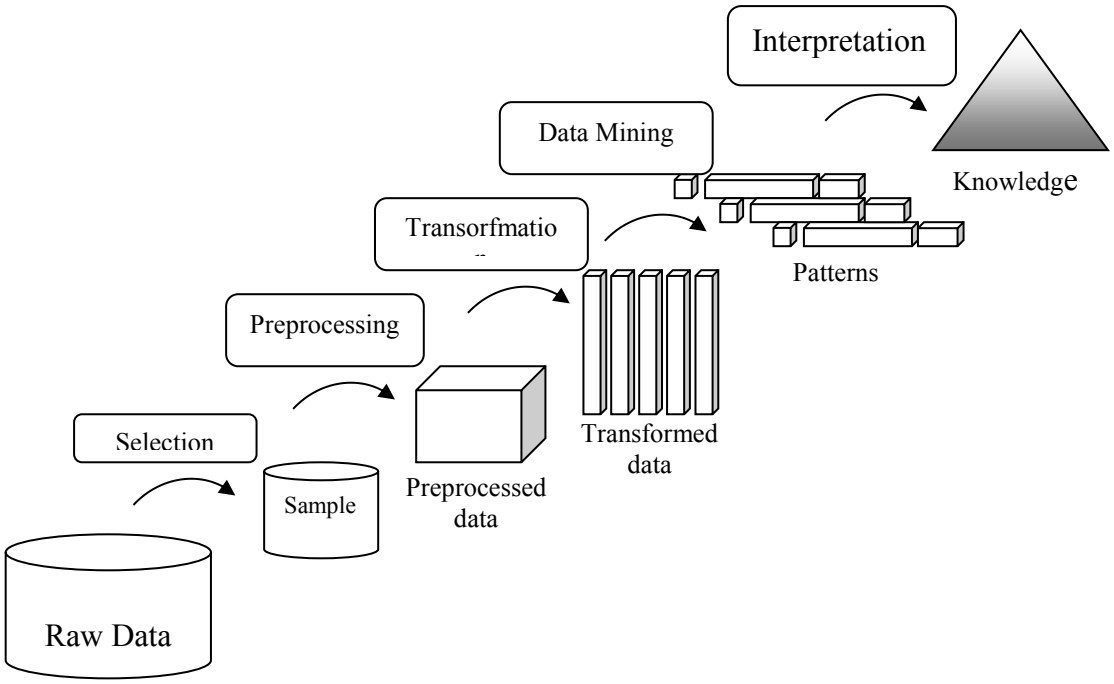



**Figure 2.2-2 KDD Process steps (source [Fay96])**

**Step 1: Selection**
The discovery process is applied on a sample from the entire volume of raw data. The sample does not suggest representation of the entire database. It is merely a stratum that allows the efficient implementation of the KDD methods. The selection of the sample is based on characteristics of particular interest of the data, for example a period of time where it is known that events of interest have taken place, or records with certain attribute values. In the context of traces for availability management, a sample can be a particular group of systems, a period of time when high failure rates where experienced or a design change was introduced.

**Step 2: Preprocessing**
The sample contains raw data, i.e. the data are in their original form as they have been recorded. Raw data usually contain uninteresting or unwanted entries, missing entries etc. Preprocessing is an important step in the knowledge discovery process. Preprocessing prepares the data set for the more information generating steps of the process that will follow. By removing unnecessary or unwanted entries and enhancing the structure of the data of interest the raw data are put into a form that will allow next steps to be more effective. During preprocessing, fundamental data structures such as distances, orders, dimensions etc. remain unchanged.

**Step 3: Transformation**
The data transformation step is perhaps the most important step of the whole KDD process. This step of the transformation process is the most relevant step for this thesis. The goal of the transformation is to increase the informativeness (reducing the data size) of the data i.e. the number of data points required to convey the same amount of information. The increase of informativeness in the data set is

achieved by reducing the dimensionality of the data and by retaining the most interesting features by which the data can be represented [Fay96]. The transformation step of the KDD framework has similarities with signal processing in the sense that the desired information in a signal is extracted by enhancing the interesting features and weakening the irrelevant or noisy features [Ben80].

Reduction of the dimensionality is the operation where several data points can be replaced by objects that represent more interesting concepts. Meaningful concepts can arise from raw data with the help of abstraction i.e. the representation of one or more data instances by familiar abstract concepts. Conceptualization increases the effectiveness of the knowledge discovery process by allowing the latter to operate with objects that are meaningful in the domain. The same time conceptualization should ensure that the objects retain their connection to the data set. Last but not least conceptualization should be done in respect to the data analytic method that will be used later for knowledge discovery.

The above is in line with the objectives of this research, namely to find and replace subsequence with point representations without losing relevant information.

**Step 4: Data Mining**
Data mining is the process step that enables the discovery of knowledge by searching for patterns, associations etc. Discovered patterns are often represented by models. Choosing the right model representation is as important as discovering patterns because it allows the fidelious modeling of the data [Fay96]. This step of the KDD relates to the analytical tools that can be used for availability management such as the discovery of association rules.

**Step 5: Interpretation**
In the last stage of the discovery process the knowledge that is discovered is being interpreted and used either directly in an application or as an input to promote further new discoveries.

### 2.4.2 Elements of KDD

Next to the framework there are a number of methods and tools that play a key role in the knowledge discovery process. Some, like data visualization techniques, help gain better understanding of the problem in hand by allowing intuitive exploration. Others, like domain knowledge, play a supportive role by setting the premises for the formulation of hypotheses. Different forms of reasoning help promote knowledge discovery by extending existing knowledge to the new data, testing discovered patterns against known facts and redirecting the search. These elements are discussed in detail below:

#### 2.4.2.1 Data visualization

The inquisitive role of the user in the discovery process is essential. Visualization of data or patterns is a strong tool in integrating the user in the discovery process. It allows the user to comprehend data quickly and to detect structures that can be exploited. The choice of the right mining tools depends very much on the effective visualization of data. Data visualization can also help extend our knowledge in the problem area and can help to form new hypotheses based on observations. For

example, data visualization may reveal unexpected patterns in the data that require new hypotheses to explain the phenomenon that created these data entries.

#### 2.4.2.2 Domain knowledge

Knowledge on a domain can be acquired during previously completed knowledge discovery processes. Knowledge can also be based on long established facts of the domain or even come from other domains that relate to the domain of interest. Domain knowledge is used to increase the efficiency of the process by allowing the analyst to emphasize his attention on the most interesting aspects of the data, to use tools that are specialized for the task and to evaluate discoveries from a better informed position [Mat93]. Domain knowledge can however become restrictive by setting tight boundaries in the discovery process. This can occur when domain knowledge is drawn from very specific cases. To avoid the restrictive effect of domain knowledge on the discovery process and to make it also applicable to a wider range of problems, domain knowledge has to remain as generic as possible [Djo95]. Domain knowledge should not act as a prejudice that can lead to the exclusion of interesting findings in the data set.

#### 2.4.2.3 Reasoning methods in KDD: Inductive, deductive and abductive

Any process that aims on knowledge discovery has to make use of one or more of the reasoning methods: induction, deduction and abduction. In KDD deductive and inductive methods have been integrated in several frameworks [She94] [Sim96] [Gre01].

Deduction, also known as the top-down approach in KDD, allows the refinement of concepts already known to the user. Concepts can be expressed as relations between data attributes in the form of "*If* A *then* B". Such relations can describe a causal relation between traces, structural information of the data sequence or simply associations between traces. Domain knowledge is a contributor of deductive reasoning in KDD.

Inductive methods or bottom-up, are purely empirical knowledge discovery mechanisms. In KDD they are used to discover rules that characterize a database. Induced rules can then be evaluated for their parsimony, generality and statistical significance [Sim96].

Abductive reasoning is less common in KDD approaches, but nevertheless a powerful way of reasoning that can lead to the discovery of new knowledge [Pei34/60]. This power derives from the ability to suggest a plausible hypothesis for an observation. Abductive reasoning can be used to express new hypotheses when an unexpected event is observed [Ho94]:

*The unexpected phenomenon* B *is observed*
*If* A *were true*, B *would be a matter of course*
*Hence there is reason to suspect that* A *might be true*.

Although at first abduction might seem to allow the formulation of arbitrary hypotheses in order to explain a surprising event, its ability to come up with new explanations should be in accordance with other background information and not just the event per se. Pavola [Pav04] discussed the role of *strategies* in abductive

reasoning and states: "*...my explanation must explain or at least be consistent with, most other clues and information that I have available concerning the subject in matter*".

The ability to discover new knowledge relies on the combinatory use of the above reasoning methods.

#### 2.4.2.4 Machine Processing

The processing of large amount of data in any of the KDD steps requires the use of machine processing. Machine processing can exploit the format of the data and perform operations of large amounts of data in close to real time manner.

#### 2.4.2.5 Unsupervised learning

In KDD the aim is to discover unknown patterns in the data that have informative value. Because the patterns are at the start of the process unknown, it is not possible to use methods that require previously training. In contrast to supervised learning where the algorithms are trained on predefined data structures KDD relies on the use of unsupervised learning methods that try to detect "sensible" structures in the data. When interesting formations are discovered they are compared to each other to reveal similarities and differences. Based on these comparisons, conclusions can derive that will lead to the definition of patterns [The06]. Because these patterns are unknown a priori they have to be validated for their plausibility. This can be done with the use of internal validation criteria [The06].

### 2.4.3 Adjusted KDD to be used for traces

The framework that is defined for the transformation of traces follows the general model of KDD (2.4.1). The KDD framework is adjusted to the particular characteristics of traces as data form and the objectives of the research. In order to distinguish between the steps that are within the interest of this research and the steps that are beyond the scope, the general framework is divided into two phases:

1. **KDD phase one** (Figure 2-3a), which is the main focus of this thesis, is responsible for the selection, the preprocessing, and the transformation of the raw data. During this phase the size of the traces is reduced without the loss of essential information. This phase is covered by chapters 3,4,5,6 and 7.

2. **KDD phase two** (Figure 2-3b) is the knowledge discovery phase. Knowledge discovery in the context of availability management in this thesis is perceived as the identification of unavailability bottlenecks, the relationships between failure and recovery events and so on. Although this phase is out of the scope of the research, examples are provided in Chapter 8 to illustrate how the transformed data obtained from phase one can be used for availability management.

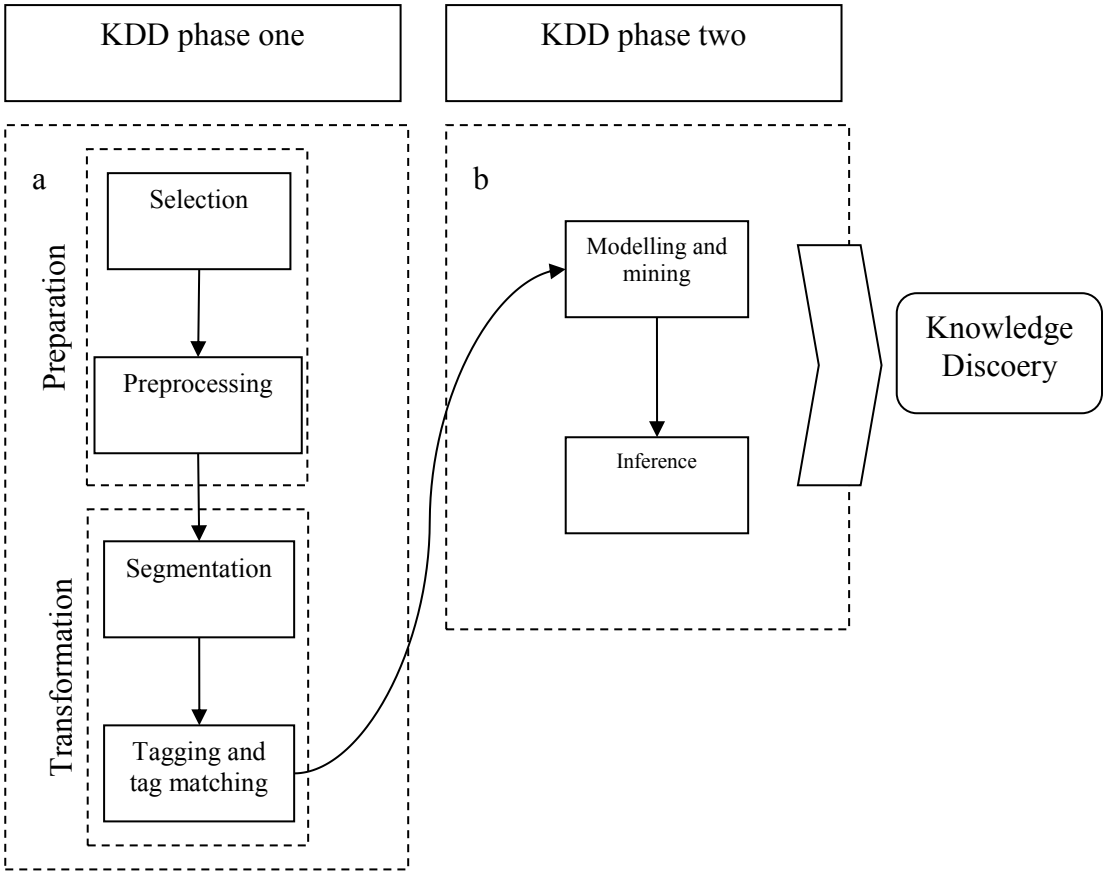



**Figure 2-3 End-to-end transformation process**

Phase one consists of two stages. The first stage is the *Preparation*. Preparation is made out of two steps: *Selection* and *Preprocessing*. The second stage is the *Transformation*. This phase consists of the steps *Segmentation*, *Tagging and Tag matching*.

### 2.4.4 First stage, Preparation

**Selection**

Raw traces are collected from the systems operating in the field. Traces recorded by the system over many hours of operating time are stored into a file. The file is remotely downloaded from the system via remote connection and stored in a database. This process is repeated over time in regular intervals so that multiple files of the same system are stored. The database contains files from several systems.

The selection of the stratum of interest is based on several factors. First is the system type. Traces of systems that use the same software version are easier to combine in analysis as the semantics of the traces are identical. Another factor is the connectivity level of the system to the network. The connectivity level of a system affects the completeness of the data set. Systems with high connectivity provide completer data because high connectivity levels provide longer periods of uninterrupted data recordings.

**Preprocessing** Raw traces contain a variety of information relating to the operation of multiple components in the system. This methodology of this thesis focuses on error and recovery traces. Before applying any type of clustering algorithms onto the traces, all irrelevant traces are removed using simple filtering methods. Preprocessing also includes the formation of one long sequence of traces out of multiple shorter sequences obtained by the log files of a single system. The concatenation of multiple sequences to form a single sequence is done with the help of the timestamps.

One of the most important aspects of preprocessing is *exploration*. The exploration of the traces helps to increase the knowledge about the structures that are found in the traces. Unknown characteristics of the traces can be discovered. These discoveries can then help to guide the development of methods for the transformation stage. In this thesis three different methods are used to explore traces:

1. Expert Knowledge:

Expert knowledge can be used to place the semantics of single traces into the context of system availability i.e. try to link the description found in a trace to the particular error that caused it and to the state of the system given that error. If such linking is possible, the perceived informativeness of traces can increase considerably. With such a linking in place, a sequence of traces can be interpreted in terms of system states and root causes that can be used directly to guide decision making. Engineers who have deep knowledge of the system's design and operation are asked to provide this link out of the context of system operation. However, interpreting of semantics using the information found in single trace and out of the context of system's operation is not an easy task. The knowledge that is obtained from this exercise can be uncertain. A method is defined that allows the direct assessment of the level of uncertainty involved in the categorization of traces, by a single expert. The method is used on a sample data set and the results are presented. The method and case study are discussed in chapter 3 in section 3.1.

2. Experimentation

Experimental settings can help understand the causal relationship between types of faults and the generation of error traces. Repetitive injections of the same fault types generate sets of subsequences that can be examined for their characteristics. Characteristics of interest are semantics e.g. which ids appear in the subsequences and quantitative e.g. the number of traces that form the subsequences, the duration of the subsequence or the time lapse between consecutive traces in the same subsequence. The information retrieved from the experimentation can help understand how the structure of subsequences depends on the type of faults. The experimentation also looks into subsequences that result from system recoveries. The knowledge obtained from the experimentation becomes part of the domain knowledge that assists the development of methods and tools in the transformation stage. The experiment and the findings are presented in chapter 3 section 3.2.

3. Graphical representation:

The graphical representation of the sequence can facilitate the exploration process by enabling the visual inspection of sequences. If visualized, complex relationships in the data such as the temporal proximities of traces in the sequence can be explored and intuitively understood. Visual inspection can lead to the discovery of unknown data structures in the sequence. Discovery of "unwanted"

structures serves the preprocessing. The removal of unwanted structures can increase the efficiency of the discovery process. Exploration of traces should be quick and effortless to allow fast iterations. A method for visualizing traces quickly and effectively is presented in chapter 3, section 3.3.

Last but not least, part of preprocessing is the removal of unwanted data structures. Unwanted structures in sequences of traces are known as partially periodic subsequences (pps). Because these structures are formed by traces of the class attribute "Error" or "Recovery" they can slip through the first naive filtering that is done during the selection step. The specifics of pps and a method for detecting pps efficiently in sequences of traces are presented in chapter 3, section 3.4.

### 2.4.5 Stage two, Transformation

As described before the aim of the transformation process is to reduce the size of the data and increase the informativeness of the data set by retaining only the most relevant features and help them to stand out with appropriate representation.

**Segmentation**: A sequence can contain numerous subsequences, each representing a physical failure or recovery event. In a raw data sequence, it is not explicit how traces belong to the same subsequence. Traces need to be assigned to subsequences on the basis of criteria that justify that assignment. The process of assigning traces to subsequences is called segmentation. As the raw data sequence can contain hundreds or thousands of traces, an automated method is needed that can assign traces to subsequences. The decision whether one or more traces belong to the same subsequence, is made on the basis of the temporal distance that separates any two consecutive traces. An unsupervised clustering method is presented that segments a sequence into subsequences. The segmentation is guided by a measure that rates the segmentation result based on the compactness of the subsequences. The segmentation method is extended to account for variation in the temporal location of traces. This method is described in chapter 4.

**Tagging:** The transformation of the data continues with the definition of point representations for the discovered subsequences. The step of the transformation process responsible for creating point representations out of subsequences is the *tagging.* The *tags* are constructs that contain all relevant semantic information of the original subsequence and but only a *single* temporal location. With the tagging the sequence is transformed to a series of ordered tags. The *tagging* step is described in chapter 5, section 5.1.

**Tag matching**: A sequence can contain several tags, each tag representing the instance of a physical event. Some of the tags in the sequence can represent instances of the same type of physical event. To reduce the size of the data and to increase the informativeness of the sequence even further it is useful to group the tags into *tag types,* where each tag type can represent a physical event type. This operation is called *tag matching*. With the tag matching, the sequence consists of representations of instances of physical event types rather than representations of event instances.

Variation of subsequences can affect the tag matching operation. In some cases tags are identical and they can be easily grouped into the same tag type with simple string matching techniques. In other cases, differences in the semantics of the tag make the

comparison and categorization less straightforward. Evaluating which tags are similar enough to be categorized to the same tag type is done using an unsupervised learning method that is based on appropriately defined similarity criteria and the semantic information found in the tags. The unsupervised learning method tolerates variation in the type of semantics and the number of semantics found in tags. The clustering of tags into tag types is guided by a measure that rates again the compactness of the produced clusters. This method is described in Chapter 5 section 5.2.

### 2.4.6 KDD phase two

In phase two, analytical tools are applied to generate knowledge from the discovered patterns. In this thesis there are two objectives in that fall are part of this phase. The first objective which is also within the scope of the thesis is to make the transformation process more efficient by using the data and the discovered patterns from multiple identical systems. The second objective, which is out of the scope of this thesis, is to use the patterns found in the traces to support availability management using analytical tools such as availability modeling and association rules.

**Characteristics of subsequences from multiple distributed systems**
The Segmentation and the tag matching of the transformation stage are both based on unsupervised clustering methods. These methods require parameterization which is performed on sampled sequences from the systems. The parameterized algorithms are then used to perform the transformation in real time manner on the same systems as they are operating in field. This means that for every newly installed system a period has to be awaited for a sequence to adequate length to sampled, before any application can begin. However given that these systems share the same design and are used in the same type of applications, it can be possible to define for each type of system a set of generic parameters that can be used for the algorithms. In chapter 6 a method is proposed that examines whether a generic parameter values can be defined for a group of similar systems (chapter 6).

**Availability management using traces**
At the end of the transformation the sequences consists of tag types. Tag types are handled as representations of failure and recovery event types. Availability modeling can be used to determine which failure types contribute most to system unavailability. The relationship between failures and recovery event types can give insight in how the system is capable in managing different failures types. The information obtained from the analysis is used to support decision making. An example of how knowledge discovery based on the transformation of traces can take place is given in chapter 8.

### 2.4.7 Data Compression

One of the main objectives of the methodology is to reduce the amount of data representations in the sequence without the loss of relevant information. The compression ratio is defined as:

$$compression\ ratio = \frac{number\ of\ \ data\ representations\ after\ transformation}{number\ of\ \ data\ representations\ before\ transformation}$$

The compression ratio is given separately for the two steps of the transformation process:

**1. Compression achieved by segmentation**

In the preprocessed sequence each single trace represents potentially one physical event. With the segmentation, all traces are organized into subsequences. The information regarding the occurrence of physical events is now conveyed by the subsequences. An equal or lower amount of subsequences compared to single traces is used to convey this information. The segmentation can therefore achieve data compression. The compression ratio is measured as:

$$cr_S = \frac{number\ of\ subsequences\ \text{in}\ the\ sequence}{number\ of\ traces\ in\ the\ sequence}$$

**2. Compression achieved by tag matching**

After the segmentation each subsequence represents an instance of a physical event. With the matching operation tags are grouped into tag types according to their similarities and each tag type represents an instance of a type of physical event. The matching operation makes the transition from a sequence of instances of unique physical events to a sequence of instances of unique physical event types. The tag matching can therefore achieve data reduction that is defined as:

$$cr_M = \frac{number\ of\ tag\ types\ \text{in}\ the\ sequence}{number\ of\ tags\ in\ the\ sequence}$$

## 2.5 Data set used in case study

To study the traces a data set collected from operational professional systems (X-ray scanner) is used. Traces are obtained from of 137 systems. The systems, of which traces are used, are selected based on a set of criteria (selection step). The sample criteria are:

1. System type: All systems in the sample are of the same type. All sampled units have the same hardware and the same software version installed on them.
2. System usage: All systems in the sample are used in the same medical field. Therefore they have comparable operational profiles.
3. System connectivity: To ensure that the traces in the sample set are representing a continuous period of time of system operation, systems with a connectivity rate above 90% were selected.

As part of the preparation phase for each system multiple log files, are concatenated to form one long sequence of traces. To concatenate multiple sequences, the temporal information of the traces is used. This single sequence should represent a continuous period of system operating time. To achieve this, two operations are performed:

1. Start-up and shut-down periods are removed from the sequence using an appropriate attribute (mode) field in the trace as filtering criterion. The reason why these periods are removed is that during system start-up and shut-down, error traces can be logged because sensors detect communication errors as some components try to communicate with others that have not started up yet. The same happens during shut down.

2. Field servicing periods are removed from the sequence. Servicing periods contain a high number of error and recovery traces due the activities of the service engineer. These periods can be removed by filtering out segments of the sequence where traces have the value 'service" in the attribute field *mode*. Also short periods of normal operation mode found between periods of service mode are removed because there is high change that the service engineer was testing the system.

The above operations produced 137 long temporal data sequences each for every system in the sample. The length of these data sequence varies between 200 and 4000 operating hours with an average of 800 hours of operating time per system. The sample data are used to apply and assess the methodology. The case study is presented in chapter 7.

# Chapter 3

# 3 Exploration and preparation of system traces

To understand how the system traces relate to and represent the physical events, it is interesting to explore the nature and the origin of traces. For the exploration, different types of methods are used to help increase gradually our understanding over error and recovery traces, subsequences and sequences. This chapter presents a series of methods that raise the level of understanding on the information found in the semantics of traces, the formation of subsequences and other data structures found in sequences.

The chapter consists of four sections:

- 3.1 Uncertainty in expert interpretation of semantics
- 3.2 Exploring the structure of subsequences with fault injection
- 3.3 Exploring long sequences of traces using data visualization
- 3.4 Data preprocessing: removal of partially periodic subsequences

Each section is self-contained i.e. it has its own problem definition, methodology a case study and a discussion. The findings from each section are used as input for the following chapters. All exploratory studies conducted in this thesis provide some new insight into traces that is helpful for the transformation stage.

## 3.1 Uncertainty in expert interpretation of semantics

One of the main objectives is to reduce the data size and make long sequences of traces easy to be interpreted by engineers. Such an objective can be met if the abstract descriptions of traces in the sequence can be replaced by other representations that contain some degree of interpretation in them. For example the subsequence in Table 2-3 can be replaced by another representation with the description “Archiving job”. The new representation contains some interpretation of the traces in it, since the information in the three traces has been combined to conclude that a file has been archived. Similarly error or recovery races can be replaced by meaningful representations. A meaningful representation is defined by the informative value the representation has within a certain business or engineering context.

The interest to replace traces by more meaningful representations was motivated by other studies [Sim05] [Mor90] [Kal99] where single traces are linked to specific fault types with help of domain experts, making their interpretation easier. In these studies various types of faults of components can be singled out with a quick inspection of the traces because the interpretation of single traces has been done in advance. For this thesis the context is availability management and for the purpose of this study a meaningful representation of an error trace would have to describe the root cause i.e. the physical event that triggered the trace, and the system state i.e. the effect of the physical event on the system. Such *a priori* interpretation of single traces in those two lines can be seen as effort to enrich the sequence with context specific information. A sequence of traces that can be replaced with such representations can be used for example to assess directly system performance with measures such as mean time between system failures, or to detect the most frequently occurring error causes.

To enable such a representation single traces need to be interpreted by domain experts. In the above mentioned studies the systems are of relatively low complexity e.g. a server that has 4 to 5 main modules. For such systems the certainty of the domain expert e.g. the computer engineer, in interpreting single traces is high, because a one-to-one mapping of traces to fault types is guided by a simple mental model of the system's architecture. Also, a relatively small scale failure mode and effects analysis could support such an effort. As the number of modules and the complexity of the system increases, cause and effect relationships become less straightforward and the linking of traces to fault types and system states becomes less accurate, hence the interpretation of a single trace becomes more ambiguous. As the ambiguity of the interpretation increases the added value of trying to enrich the sequences with domain knowledge is decreasing.

In this thesis the process of interpreting traces to representation of root causes and system states is referred to as *conceptualization*. For the conceptualization of traces a classification scheme is used where fault types and system states are represented by categories. The linking is performed by experts who classify each trace into the categories they believe are most relevant for that trace. Each trace can be assigned to multiple categories of fault types and system states. The count of category membership (single vs. multiple) measures the amount of uncertainty the expert is introducing when classifying a single trace. To capture the expert's uncertainty a confusion *measure* is defined. The confusion measure is capturing the uncertainty of a

single engineer conceptualizing a single trace. The proposed method can be extended to combine measure from multiple experts, but this possibility is not examined here.

First the conceptualization process is described in 3.1.1. In 3.1.2 the confusion measure is described. Finally a case study is presented in 3.1.3 where the conceptualization is applied on a collection of semantics of traces. The results of the case study are provided in 3.1.4 and a discussion on the findings and the conclusions are given in 3.1.5.

### 3.1.1 Conceptualization of traces

Concepts are abstractions that derive from existing knowledge models, which describe adequately the domain they represent [Gai93]. According to the prototype view [Ham93], a concept can be defined by a set of attributes that describe the concept adequately. In the context of availability management the concept *error* can be defined by the attributes *root cause* and *severity.* Root cause describes the trigger that causes the error and severity the effect the error has on the system.

To conceptualize an error trace, the engineer has to use the semantics in the trace, to indicate the root cause and the severity that the trace can represent. A classification scheme is used to facilitate the conceptualization. The attributes (root cause and severity) are represented by the classes $C_i$ of the classification scheme and the attribute *values* are a set of non-overlapping subclasses $C_i = \{c_1, c_2, ...c_n\}$. Engineers assign traces to sub classes. The assignment of a trace $x$ to a subclass $c_i$ is indicated by the weight $w(x, c_i)$ [Haw81].

Concepts can be described in various levels of abstraction. The level of abstraction is defined by the values that are given to the attributes. For example for the concept *root cause* the attributes can be defined on a level such as *database connection timeout, database connection* or just *database*. The first level has more information in describing the root cause than then other two levels. The abstraction level of the concepts that is used in the interpretation defines the balance between the uncertainty and the informativeness of the interpretation. More detailed concepts can have a higher informative value but can come with higher levels of uncertainly in the interpretation.

When a trace is assigned to a single sub class the weight for the assigned subclass $c_i$, becomes $w(x, c_i) = 1$ and for the unassigned subclasses $j \neq I, w(x, c_j) = 0$. Assignment to multiple sub classes is permitted. A trace $x$ can be assigned to multiple sub classes of a class if the following conditions are met:

$$0 \leq w(x, c_i) \leq 1 \text{and} \sum_{i=1}^{n} w(x, c_i) = 1 \quad (3\text{-}1)$$

The assignment of a trace to multiple subclasses of an attribute marks the uncertainty of the expert for that assignment. The value of the weight $w(x, c_i)$ can be given by the expert based on the confidence of his assignment (less weight suggests less confidence). In the absence of weight specifications, the weight can be distributed uniformly among all subclasses in which case the weight per subclass is equal to the reciprocal of the number of subclasses assigned.

### 3.1.2 Confusion measure for the classification of error messages

The confusion measure captures the uncertainty associated with the linking of a single trace to an error attribute. The uncertainty measure $H(w)$ is based directly on Shannon's formula for measuring the entropy of random variable [Jay79] and is defined as:

$$H(w) = -\sum_{i}^{n} w(x,c_i)_i \log w(x,c_i) \quad \textbf{(3-2)},$$

Intuitively the confusion measure resembles Shannon's entropy of information in the sense that, as the increase of the number of likely outcomes of a message increases Shannon's entropy, so the increase of the likely ways to interpret a trace increases the confusion measure. The confusion measure describes the uncertainty of a single engineer when he interprets the semantic information of a trace by using a finite set of concept attributes. The distribution of the confusion measure describes the uncertainty in the interpretation of the entire set of traces. Given the distribution of the confusion measure for a set of traces, it is possible to see if the interpretation of traces by a single expert is with confidence about the meaning of the traces or not.

The measure is takes it lowest value $H(w) = 0$ (no uncertainty) when the trace is assigned to a single sub class ( $w(x,c_i) = 1$ ) and it takes the highest value when the trace is assigned to all sub classes and only if $w(x,c_1) = w(x,c_2) = ... = w(x,c_n)$ and $w(x,c_i) = \frac{1}{|C|}$, where $|C|$ is the size of the class i.e. number of sub classes. Classes with a higher number of subclasses assume higher values for maximum confusion.

To illustrate the use of the confusion measure an example is provided:
30 traces are conceptualized. Each trace can be assigned to four different subclasses of an attribute by one classifier (expert). Because the classifier has four options, the confusion measure trace that is assigned, can take four values i.e. $H(w) = \{0, 0.6931, 1.0986, 1.3863\}$ depending whether the trace is assigned to one, two, three or four subclasses respectively. In the case where the expert classifies all traces with full confidence, the distribution of the confusion measure for this set of traces has the form as seen in Figure 3-1. The confusion measure for each trace is zero, since each trace was assigned to one single subclass.

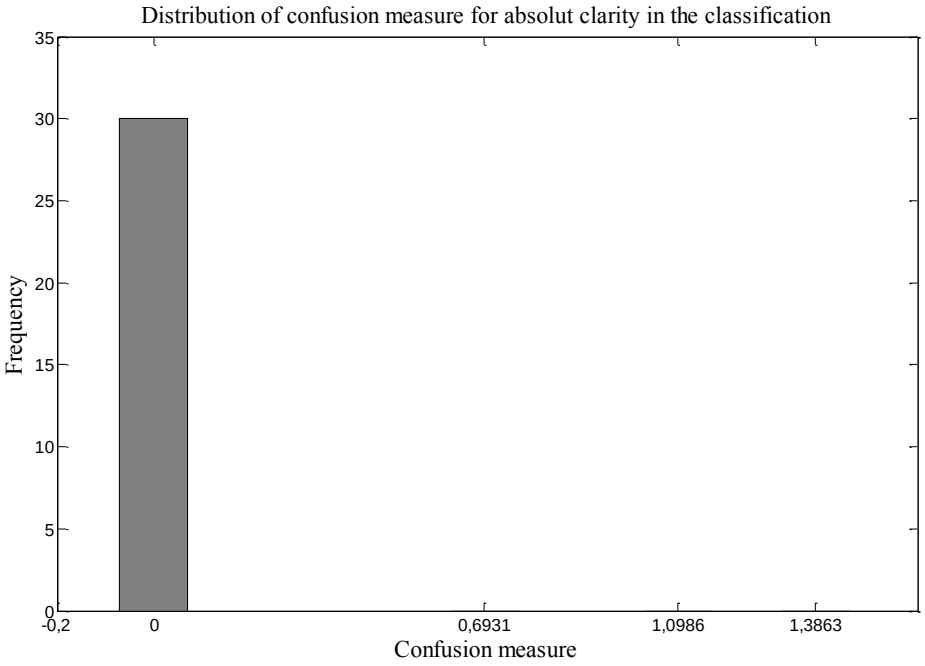



**Figure 3-1 Distribution of confusion measure for no uncertainty in the classification**

In a different scenario (e.g. other expert), each trace is assigned by the classifier to all four available subclasses, the confusion measure for each trace takes the maximum value, which for this case is equal to 1.3863 (Figure 3-2).

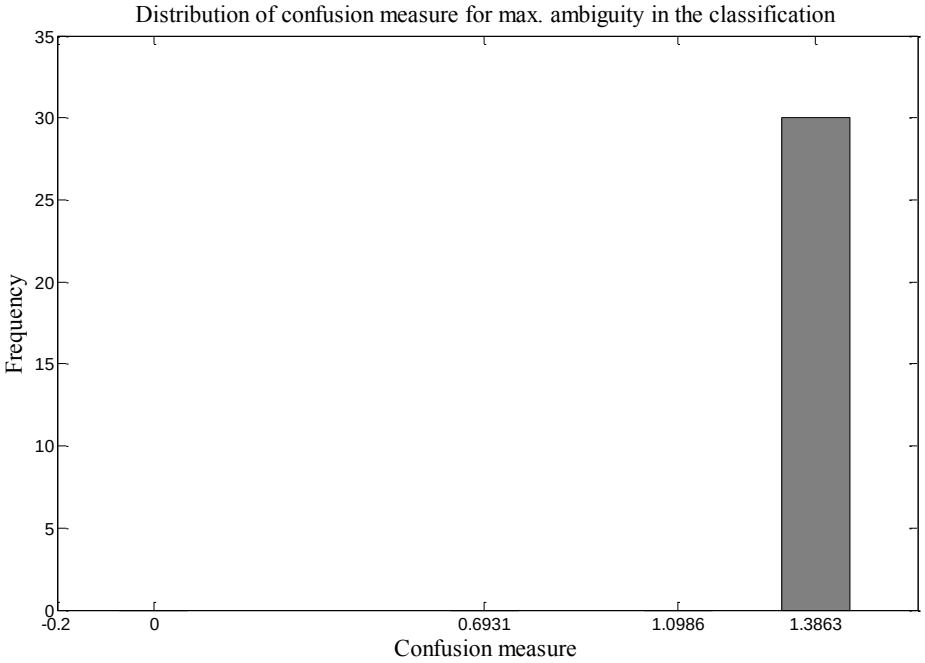



**Figure 3-2 Distribution of confusion measure for maximum ambiguity**

For both scenarios the situation, regarding the amount of uncertainty found in the interpretation of traces, becomes immediately clear by the two distributions of the confusion measure. Any amount of uncertainty lying between these two extreme scenarios can be represented by the distribution of the confusion measure. This representation is effective in conveying directly the information to the analyst

regarding the amount of uncertainty induced during the interpretation of a set of traces.

There are two advantages of using Shannon's entropy theorem as a basis of the confusion measure:
1) The measure can be generalized to use the broader framework of belief measures and plausibility measures [Kli87], which, under certain conditions, might be a more appealing method to use in expert knowledge elicitation, since these measures can account for ignorance from the expert's side.
2) The measure allows the use of Dempster and Shafer's (D-S) rule of combination [Dub99] [Sha76] to update the measurement with new evidence. The D-S rule allows as well the combination of evidence provided by several experts.

### 3.1.3 Case study: Interpreting error messages with the help of experts

A set of 416 unique error traces is collected. This is the complete set of traces that can be logged by the system of the sample data. Experts were asked to conceptualize these traces. The concept "fault type" is defined by two classes: *root cause* and *severity.*

Due to the multiplicity of the components and the complexity of the system's architecture, it was decided to abstract for both classes. The sub classes for class *root cause* are:
1) *Software* i.e. the trace represents an error for which the root cause is found in the system's software
2) *Hardware* i.e. the trace represents an error for which the root cause is found in the system's hardware

For the class *severity* three subclasses were provided:

- *Critical*: This type of error may cause a mission failure, unacceptable downtime, or loss of data. The effect of this error is disruption of the workflow by system unavailability that cannot be overcome without expert support.
- *Moderate*: This type of error may cause undesirable downtime or partial loss of function, but it may be temporarily circumvented. Error leads to temporary disruption in the workflow that can be overcome by the user by restarting the system. This error may lead to loss of metadata or limited performance like for example longer reaction times, or lower image quality. .
- *Negligible*: This type of error has little effect on the functionality of the system except that it may be a nuisance to the user. Error leads to prolongation of execution time by recalling the command. It may never be fixed and still have a negligible effect on the overall system.

The 416 traces were divided into groups according to the modules they originate from. One module expert was asked to classify the set of traces of every module. Dividing the traces based on relevant modules and asking the corresponding expert should increase the chance of classifying the trace with high confidence i.e. fewer selected subclasses.

The weights were distributed uniformly among the number of assigned subclasses of each attribute. For example, for the attribute *severity* if the trace assigned to three subclasses, each value would have a weight of 0.33.

### 3.1.4 Results of case study

The overall results of the classification process of the 416 traces, for the attribute *severity* can be seen in Figure 3-3 and for the attribute *root cause* in Figure 3-4.

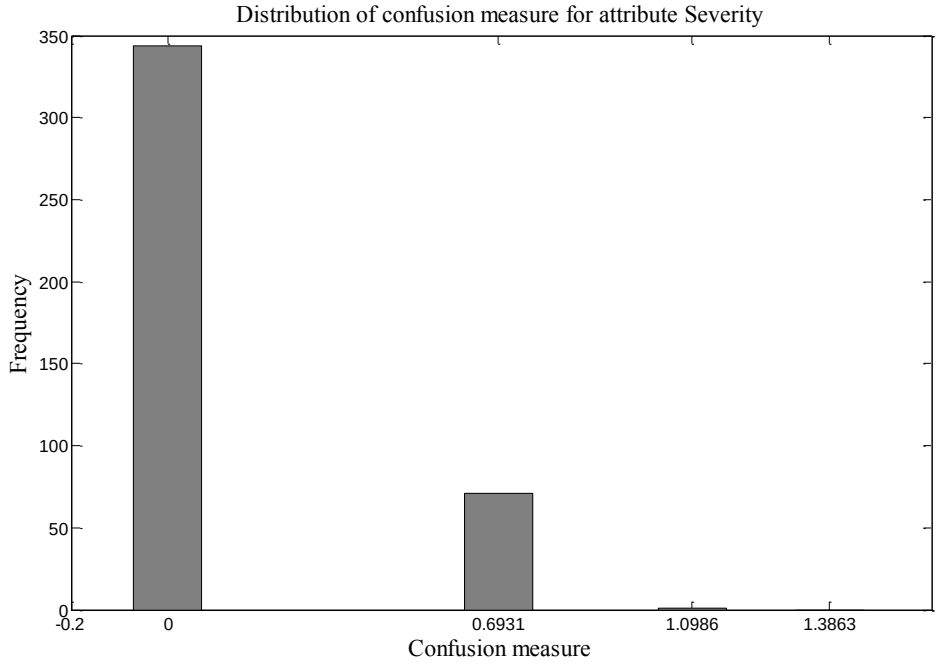



**Figure 3-3 Distribution of confusion measure for *severity***

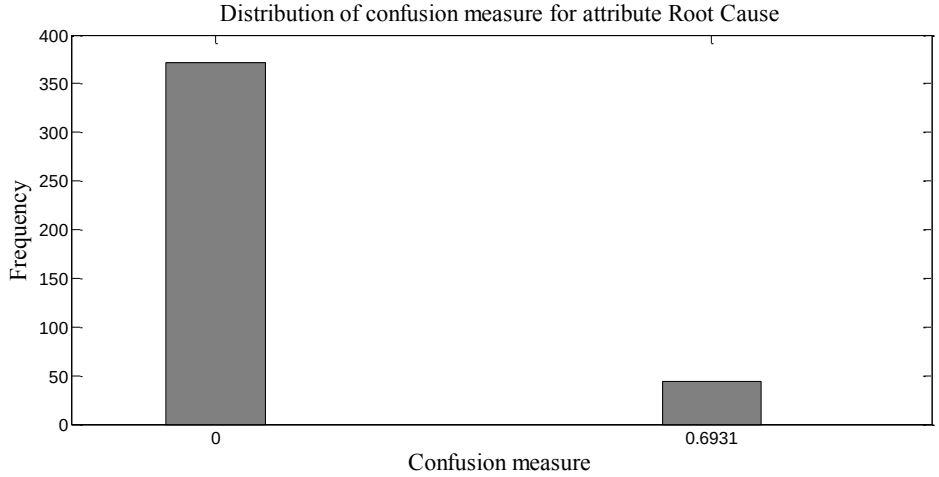



**Figure 3-4 Distribution of confusion measure for *root cause***

Both figures convey the information on the overall uncertainty of the linking for all traces in the set. The confusion measure with value zero (no uncertainty) is the highest in both graphs. However, the linking of traces to the attribute *severity* involved more uncertainty than making the linking to root cause subclasses. This is obvious by the higher bar of the confusion measure with value 0,6931 for the former. Nevertheless the amount of uncertainty measured for the assignment of traces to *root cause* is worrying, considering that the experts had to choose between only two subclasses each representing highly abstract attribute values (*hardware* vs. *software*). For certain

modules almost half of the traces were assigned to both subclasses of the attribute *root cause* (Figure 3-5).

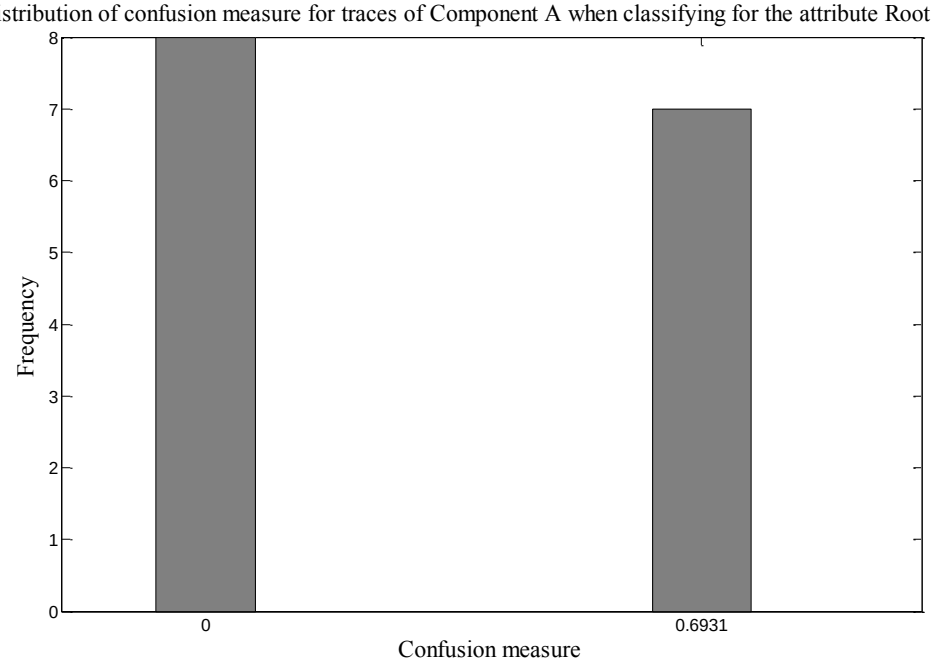



**Figure 3-5 Distribution of confusion measure for attribute *root cause* for the set of error messages corresponding to sub-system A**

### 3.1.5 Discussion and conclusion

Though the classification of the traces in this case study showed that overall the uncertainly about the linking of traces the error attributes was low, considering the level of abstraction, especially for the attribute root cause, the conceptualization has little practical value. To increase the practical relevance of the conceptualization the attributes have to be defined in higher granularity. If the attributes are defined with higher granularity e.g. specific components for root cause, the distribution of the confusion measure has to be redefined, depending on the number of subclasses for each error attribute.

This case study supports the view that the transformation of traces from single semantics to meaningful representations is difficult when done out of context. The interpretation of the trace may depend on the conditions in which the trace was logged, for example the type of mode the system was in, start-up mode vs. normal operation mode, or the type of operation it was performing when the error occurred. Instead of interpreting the semantics of traces upfront without any knowledge of the context, it is likely more effective to perform the interpretation of traces after the traces are prioritized based on their relation to system unavailability. Fewer number of traces that are highly relevant for system unavailability can be examined in depth taking in to account more information about the system state at the time the error trace was logged and by using full root cause and system effect analysis.

## 3.2 Exploring the structure of subsequences with fault injection

Subsequences are the end product of a process that begins with an error occurring in a single component and ends with the logging of one or more traces in the trace sequence. A fault causes a component to experience an error. The error might propagate to other components due to the functional dependencies, causing them also to produce errors. The errors are picked up by the "sensors" in the system. The logging mechanism reports the erroneous states in the form of traces. The collection of traces resulting from a single physical error event, form an *error subsequence* (Figure 3-6). A similar process produces the subsequences that describe a system recovery.

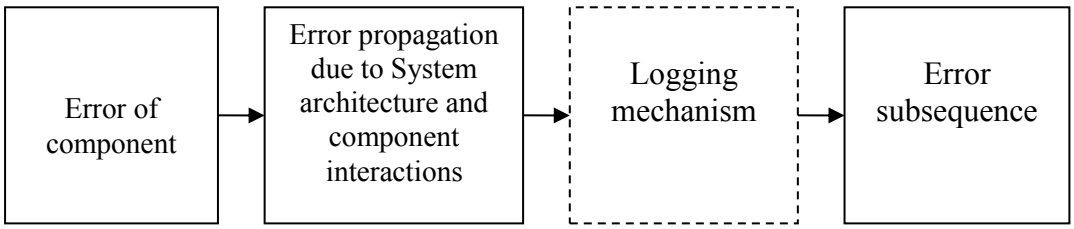



**Figure 3-6 Process from error to subsequence**

As the process repeats itself over time it results to the formation a long sequence that contains several subsequences.

To transform a long sequence of traces into a sequence of point representations for physical events, the subsequences have to be identified first. To device methods that can identify the subsequences in long sequences of traces, a better understanding of the relationship between physical events (the triggers) and subsequences is needed. To understand this relationship an exploratory study is conducted. The exploratory study is performed with the help of *fault injection experiment*ation. The experiment consists of three phases:

1. A certain fault is injected into the system multiple times under the same conditions
2. The *subsequences* resulting from the multiple injections of the same fault are collected and compared.
3. The experiment is repeated for different types of faults that are selected to cover a wide area of the system's architecture.

The exploratory study intends to address the following questions:

- Verify the causal relationship between faults and traces as it was suggested by the conceptualization in 3.1. The root cause that was indicated by the expert as the most likely trigger of a trace is now injected into the system and the resulting subsequence is searched for the trace in question. The state of the system after fault injection is also compared to the system state suggested by the expert.

- Given that the structure of a subsequence is defined by its temporal and semantic characteristics, how do these characteristics manifest in a subsequence for different types of faults?

- Do subsequences that result from the same error under the same conditions show variation in their structure or is the process deterministic? Information on the structure of subsequences e.g. the number, type, order of traces, the length of a subsequence, the temporal distance between consecutive traces within the subsequence are compared.

- Does the system recover after the failure as expected and is that recovery visible in the traces? This steps it to verify the presence of recovery traces and their association to system failures.

This section is organized as following: firstly, fault injection techniques used in other studies are discussed briefly in 3.2.1. Then the experimental setup and its execution are described in 3.2.2. A set of observation areas are defined that can help draw conclusions from the experimentation results. These observation areas are described in 3.2.3 and the results of the experiment are presented 3.2.4. Finally the experiment results are discussed in relation to the objectives of the study and conclusions are presented in 3.2.5.

### 3.2.1 Related work

Experimentation gives the opportunity to explore unknown mechanisms by carefully controlling the inputs and the surrounding conditions. It also allows problems that are too complex for analytical modeling to be studied in parts and learn about the dynamics that govern them. Fault injection is an experimental technique that is used to assess the behavior of operating components or systems when subjected to faults. The dependability of processors and processor architectures and the detectability of faults and errors has been the focus of several fault injection experiments [Mar02] [Arl90] [Con99] [Car98] [Gu03] [Cho90]. To facilitate such experiments, techniques for fault injection on hardware and on software have been developed [Kan92] [Hsu97]. These techniques inject the faults either directly into the processor's pins using an erroneous input or into the software by manipulation of the code. Most of the techniques require the use of special hardware or software to inject the faults at the right moment and at the right location during system operation. Sophisticated methods like these are required in order to test resilience of components during operation. Some of these studies examine the ability of the logging mechanism to detect errors; however none of them examine the effect of the logging mechanism itself on the traces.

In this study the focus is exploring the relationship between the physical error events and the structure of subsequences. The techniques used in this research are simpler and do not require the destruction of components. The choice for the techniques used was based on the feasibility of injection and the reproducibility of the injected faults.

### 3.2.2 Experimental set up

The system in the study is a medical imaging system. The hardware and software configuration is such, to match that of the sample of systems used in the reset of the research (see 2.5). The system was installed at the site of the manufacturer and was in full operational condition.

The inputs for the fault injection experiment are taken from the results of the conceptualization process (see 3.1.5). The link between trace, root cause (fault) and

system state is used. For a given trace the indicated root cause is injected into the system with the intention to receive the same trace back (Figure 3-7). If necessary a function of the system is initiated to trigger the error e.g. fluoroscopy is triggered. Once the fault is injected the resulting error subsequence is collected, the system state is monitored and if available the resulting recovery subsequence is collected (Figure 3-7.b). Once the cycle is complete, the system is brought back to normal state either by automatic or manual recovery. For a given trace the process of fault injection is repeated multiple times.

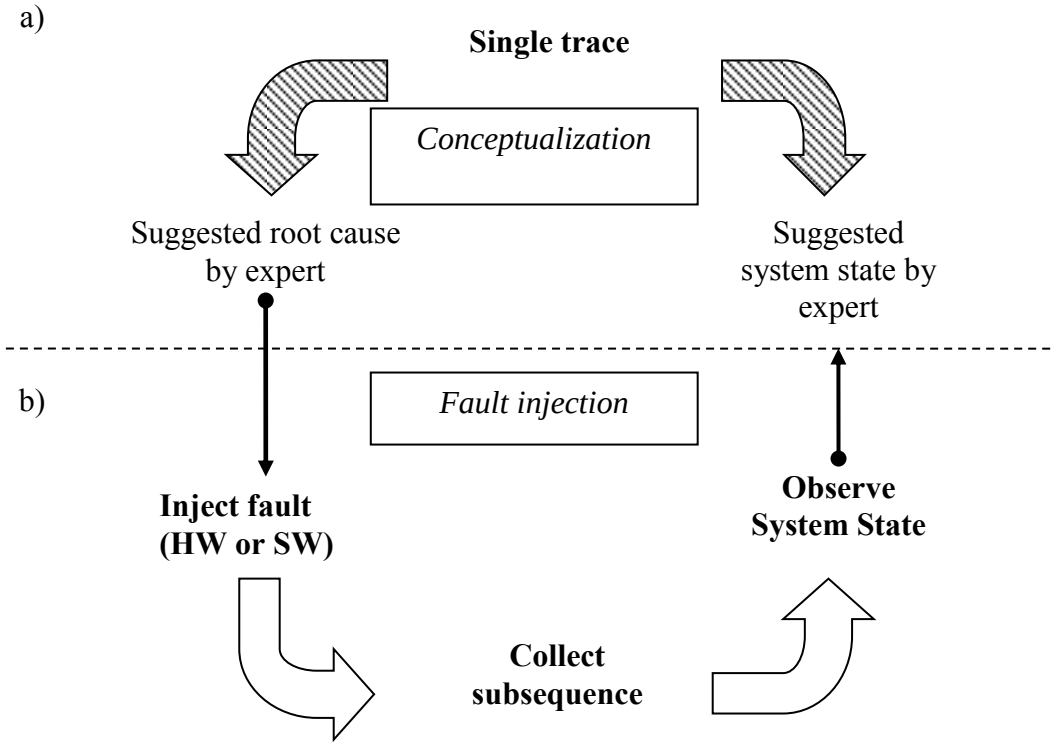



**Figure 3-7 Fault injection experiment**

A set of fault types is prepared for injection. The faults to be injected are selected based on the following criteria:

a. Technical ability to inject the fault: fault injection has to be nondestructive for the component and easy to execute. For example, to achieve communication interruption the cable connecting a component to its control board is disconnected at an appropriate point in time
b. Coverage of the system's architecture: The injected faults should cover a wide area of the system's architecture. This design allows examining whether the location of the fault has an effect on the formation of the subsequence.
c. Severity of the error: The injected faults should represent all severity subclasses as defined by the classification scheme of the conceptualization process
d. Root cause of the error: The injected faults should cover both, hardware and software components

A restriction on the selected number of faults for injection was put on the time that was available for fault injection (the system was available for two days). 21 faults were selected to be injected into the system. The set of selected faults fulfill the above criteria.

The distribution of the injected faults over the system's architecture is shown in Figure 3-8. Subsystems are indicated by the areas with the red contour. Each subsystem consists of numerous modules represented by boxes. A module contains one or more components. faults are injected into the components. The system has four architectural layers (layers are separated with blue horizontal lines):

1. User interface
2. Application layer
3. Application library layer
4. Technical layer

Usually in multilayered systems the components located in the highest layers e.g. layer 1, have dependencies on the components on lower layers. Errors tend to propagate from components in deeper layer to components in higher layers. An error that occurs in layer 4 is more likely to cause other components in higher layer to experience errors.

In Figure 3-8 the distribution of the locations in the system to be subjected to fault injection are shown. The squares represent modules. Modules that form sub-systems are grouped by the red outlines. The modules that contain components *capable of detecting errors and recording traces* are represented by the colored squares. Out of these components, the modules that contain components that are chosen to be injected with a fault are colored in black .The distribution of the black squares shows how the injected faults cover a wide area of the systems architecture targeting different subsystems of all four layers.

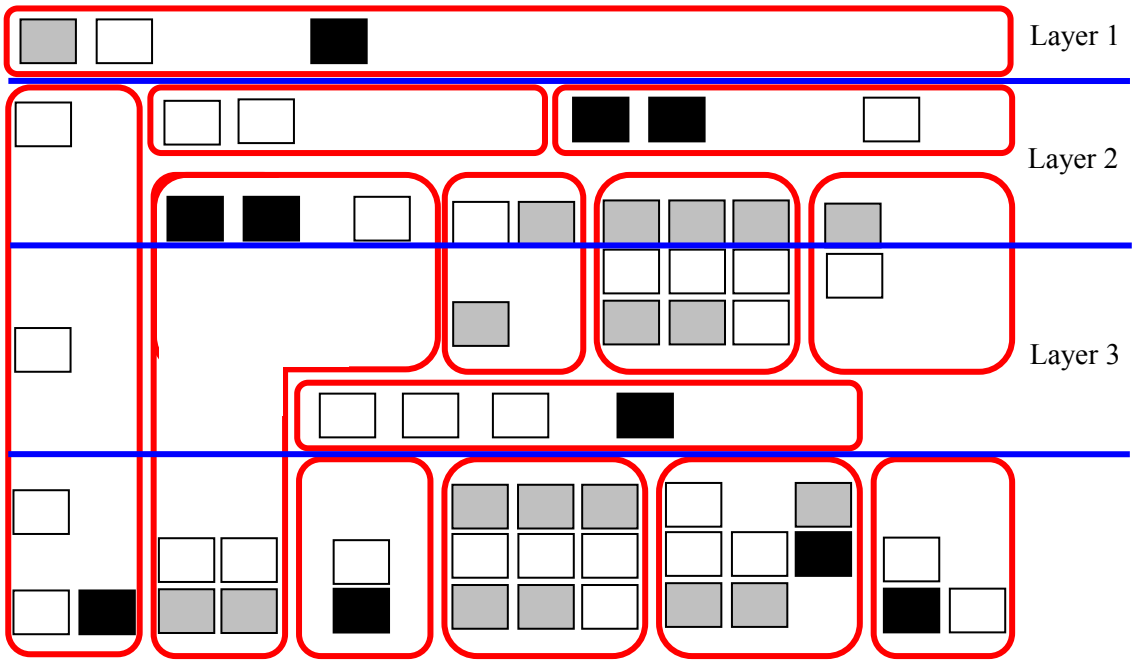



**Figure 3-8 Targeted components (colored squares) in system's architecture**

The 21 selected faults to be injected are distributed over hardware and software components and are of severity subclasses *Critical* (B), *Moderate* (C) and *Negligible* (D) (Figure 3-9, label format (severity, count)).

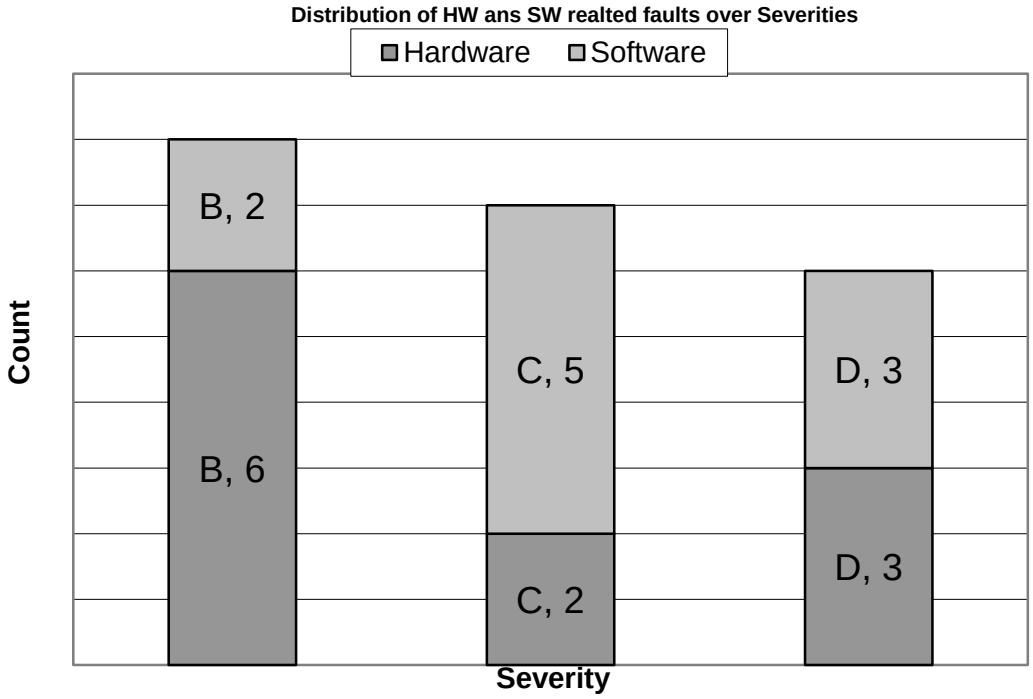



**Figure 3-9 Distribution of hardware, software faults across severities**

Methods on how to inject the faults were obtained by domain experts. These are grouped into six general categories, each relating either to hardware or to software faults (Table 3-1).

| Fault injection method | Hardware | Software |
|---|---|---|
| Disconnect cable/ remove component | 10 | |
| Terminate process | | 5 |
| Corrupt file or registry | | 4 |
| Incompatible procedure | 1 | |
| Software incompatibility | | 1 |

**Table 3-1 Methods of fault injection**

### 3.2.3 Experimentation focus areas

The experimentation focus areas are aspects of the experimentation that can help answer the predefined questions. These focus areas are described here under the corresponding experimentation objective:

- *Causal relationship between the fault and the trace*: A specific trace is expected to be observed for a specific injected fault (Figure 3-7b). When the expectation is met, the existence of a causal relationship between the fault and the trace is verified. However, if the expectation is not met, it does not necessarily suggest the lack of such relationship. Failing to trigger the expected trace may be due to missing conditions during the fault injection.

- *Structure of subsequences*: The subsequences that result from the fault injections are collected and examined. The information collected can help understand the

structural characteristics of the subsequences for different types of faults and the variation in the structures of subsequences resulting for the same type of fault.

The structure of a subsequence is described by the recording of the following characteristics:

a. The number of traces in the subsequence
b. The number of different types of traces in the subsequence.
c. The temporal distance (in seconds) between traces in the subsequence
d. The length of the subsequence
e. The order of the traces in the subsequence

The observation of the characteristics aims in comparing the subsequences that are obtained from multiple injections of the same type of fault.

1. *System Recovery*: Recovery related subsequences are the result of the recovery processes that restore the system to normal state after a failure. This experiment aims in verifying the existence of recovery subsequences and examine their latency in relation to the error subsequences.

### 3.2.4 Results of experimentation

The experiment was conducted over a period of two days. Out of the 21 selected fault injections 20 were executed. One fault type was left out because it was not be successfully injected. The complete cycle of fault injection, data collection, system recovery and system restart, proved to be more time consuming than expected (approx. 25 minutes). Because of that only limited repetitions of each experiment were conducted. Observations were made on all areas of interest. The observations are presented below:

*Causal relationship*: For 14 out of the 20 injected faults, the expected trace was observed (Table 3-2). For the rest of the injected faults, error messages that are semantically close to the expected messages were observed. The intended error messages were not observed for 4 faults injections that involved a cable or component disconnection and 2 that involved a registry corruption.

| | Intended trace present? | |
|---|---|---|
| Fault Simulation | **No** | **Yes** |
| Disconnect cable/ remove component | 4 | 6 |
| Kill process | | 4 |
| Corrupt file or registry | 2 | 2 |
| Incompatible procedure | | 1 |
| Software incompatibility | | 1 |

**Table 3-2 Observation on the causal relationship between faults and error messages**

*Structure of subsequences*: The number of traces found in the subsequences for injected faults varied from 1 to 151. There was no correlation between the number of traces found in the subsequences and the depth of the architectural layer in which the fault was injected. The length of the subsequences for injected faults varied from few milliseconds to 200 seconds. The length of the subsequence was found to be positively correlated to the architectural layers of the system i.e. faults injected in deeper layers produced subsequences of longer duration. The number of different

traces in a subsequence varied from 1 to 16. There was no sign of association between the number of traces and the number of different types of traces in a subsequence. There was also no sign of association between the number of different types or traces in a subsequence and the depth of architectural layers in which the fault was injected. The distance between consecutive traces within a subsequence varied from few milliseconds to 180 seconds.

*Variation in the structure of subsequences of the same fault type*: For this observation the fault injection was repeated several times and different instances of subsequences were obtained. Because of the duration of the fault injection procedure only two faults were selected: *Fault A* was repeated 8 times and *Fault B* 4 times

a. *Fault A*: The subsequences resulting from the injection of *Fault A* used a pool of 23 traces. In the 8 subsequences obtained it was observed that each subsequence contains from 11 to 17 (43% to 73 %) of the traces in the pool. Moreover the order of the traces in these subsequences can differ significantly. The total length of the subsequences varied from 71 to 159 seconds. The distance between successive traces in the subsequences varied as well from few milliseconds up to 60 seconds.

b. *Fault B*: The instances produced by *Fault B* showed greater consistency. The pool of traces is formed by only 5 traces. Each of the 4 subsequences obtained contained 2 to 4 (40% to 80%) of the traces in the pool. The order of the traces in the subsequences of this fault had partly a pattern. Two of the traces that appeared in all instances, appeared in the same order in all subsequences. The other traces of the pool were either absent or ordered randomly. The total length of the different instance varied between few milliseconds to 26 seconds. Similarly the distance between consecutive traces varied across the subsequences but will smaller fluctuations.

*System Recovery: Fault A* provided a good example for system recovery observation. The injection of the fault required the termination of a specific process. On the occurrence of the error the system initiates the recovery by restarting the terminated process. In all 8 injections of Fault A the recovery of the terminated process was found in the temporal data sequences following within few seconds. For *Fault B* no automated recovery was observed.

### 3.2.5 Discussion and Conclusions

The causal relationship between failures and the resulting traces was verified for most of the cases of injected faults. For 6 out of the 20 injected faults the expected trace was not observed. All traces that were selected for the fault injection experiment had no uncertainly in their interpretation (see 3.1). The result is another indication that the linking of traces to concepts with the help of experts does not provide reliable results.

The observations made on the structure of subsequences had some interesting findings. For the subsequences that were obtained from all injected faults, considerable variations were fund in the duration of the subsequence, the number of traces and the type of traces. Given that these subsequences were obtained from different injected faults, the result is not surprising. Similarly there was variation also in the distance between successive traces is the subsequences. However these distances were for most of the cases only few seconds. The largest distance between

two consecutive traces that was observed was 180 seconds. The distance between the successive traces in a subsequence that was observed in the experiment is very short considering that successive subsequences can be separated by many hours of operating time. This finding suggests that subsequences are dense data structures in the sequence can span over hundreds of hour of operating time.

Another interesting observation was made on the subsequences obtained from the multiple injections of the same fault. In both cases (*Fault A* and *Fault B*) the subsequences produced by the same fault, varied in the number of traces, the number of different types of traces and their order of appearance. It is clear that the concept of *fault signature*, a subsequence that is used as an identifier of a specific fault, applies loosely here because patterns do not appear in a deterministic way. Subsequences that are produced due to the same fault are found to have variation in their structure. Variation can be found in the temporal and the semantic aspect of their structure.

The experiment provided also the first evidence on the association between error and recovery subsequences. The latter were observed after a relatively fixed period of time that was defined by the duration of the system recovery process.

## 3.3 Effective visual representation of traces for fast exploration

So far the information collected on traces is obtained from field studies, first the conceptualization processes with the help of system engineers and then the fault injection experiment in a controlled environment. In both cases the knowledge that is gained over the traces is based on the assumption that the system operates in generic or constant conditions. The experts interpret traces given a limited number of generic scenarios of system operation. The experimentation on the other hand provides a view on subsequences under controlled and constant conditions. Systems that are operating in the field experience a wide spectrum of conditions. Given that, sequences that are collected from systems that are operating in the field are likely to contain a bigger variety of data structures.

To utilize these sequences in order to increase the knowledge on subsequences of traces, a method is needed that will allow fast exploration of sequences obtained from system operating in the field. Such an exploration can be performed with the help of visual representation techniques. Traces can be represented visually to effectively reveal both, the semantic and the temporal characteristics of the subsequences. Visual representation as an exploratory technique has its advantages. The visual representation allows the human to intuitively explore the data. In cases where little is known about the problem in hand, visual representation can help with its fast learning iterations to adjust the objectives of the data analysis [Kei01].

To facilitate the semantic and temporal information, traces can be plotted on a plane where one dimension (vertical) represents the semantics and the other (horizontal) represents time. A sequence can contain multiple semantics. Their arrangement on the vertical axis can be done either randomly or controlled. Generally in visualizations, randomly positioned dimensions yield less information than ordered dimensions [Ma99]. The ordering of the data has to be done accordingly to enhance certain features of interest. To achieve effective visualization of traces it is preferred to order the arrangement of semantics of the vertical axis of the graph according to their pairwise associations.

In this thesis the item to item similarities are assumed to be defined by the underlying functional dependencies of the components of the system. When an error occurs in a component multiple types of traces can be logged closely to each other to form subsequences. The multiple traces are logged as the error propagates to other components of the system that are functionally coupled with the first erroneous component. The stronger the coupling between the components the more consistent the co-occurrence of the traces is. This co-occurrence of traces, referred to as *association*, is a manifestation of the functional coupling of components. An appropriate measure is used to capture the association of traces and use it to enhance the readability of visualization of traces.

Various techniques, such as multidimensional scaling, factor analysis, principal component analysis, are available for enhancing exploratory data visualization [Fer03]. Among them, multidimensional scaling is often used in data visualizations to enhance functional dependencies [Kei96]. Metric multidimensional scaling uses a matrix of item to item similarities to optimize their positioning on a low dimensional space, so that the distances between the item to item positions agree with the item to

item similarities. Defining an item to item similarity measure that can be used in fast visual exploration of traces is the main interest of this section. The optimization method itself is out of the scope of this thesis.

This section is organized as follows: Section 3.3.1 describes the problem of non-ordered visualization and sets the basic requirements for effective visualization. In section 3.3.2 a framework borrowed from the domain of multidimensional scaling is described with which dimensions in a graph can be ordered to enhance the features of the data. For traces the feature of interest is their association in the sequence. In section 3.3.3 a measure of association is described that is appropriate for traces. The method for measuring association of traces and the resulting artifacts needed for the scaling operation are described in 3.3.4. In 3.3.5 the scaling operation is described briefly and the effect on the visualization is discussed. A case study is presented in 3.3.6, where the effective visualization is used to explore traces collected from systems operating in the field, with the intention to increase the understanding on the structure of subsequences. Section 3.3.7 contains the discussion and conclusion on the effective visualization and the finding of the exploration.

### 3.3.1 Visualization of traces

To create a graph of traces that can convey the temporal and semantic information found in the sequence, a two dimensional space is needed. Time is represented on the horizontal axis and the types of semantics represented on the vertical axis. In Figure 3-10 an example is shown of a sequence plotted on such a two dimensional graph. In the sequence there are 32 different types of semantics. These are positioned on the y-axis in a lexicographic order since no other ordering is suggested at this point. For the plot of the graph each trace (represented by a red cross) is positioned using the coordinates *trace ic* (*ic* represents uniquely the semantics) and *time stamp*.

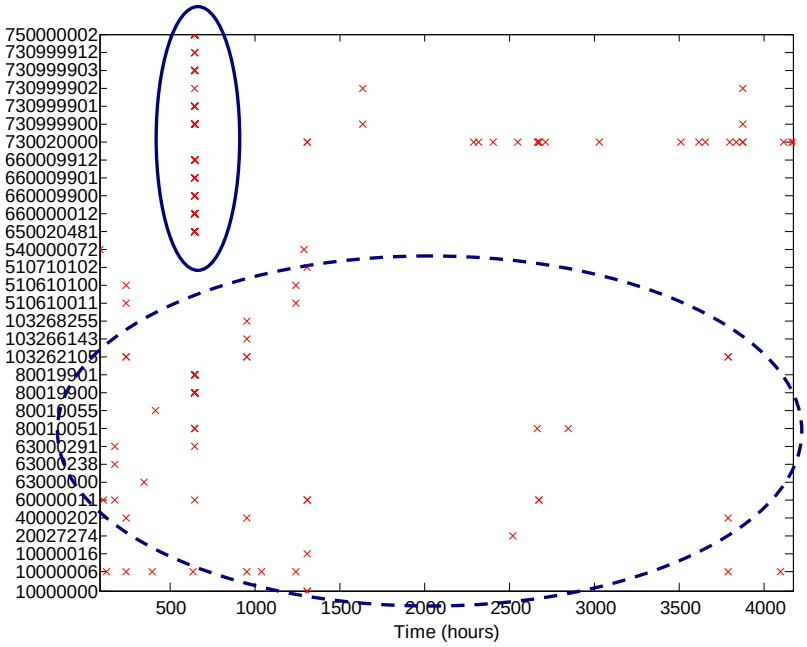



**Figure 3-10 y-x plot semantics vs. time**

The length of the sequence is over 4000 hours of operating time. The traces in subsequence are separated by few seconds. The distance between successive traces in the subsequence is very small compared to the length of the sequence. This makes the

traces that are close to each other appear on the graph as if they are arranged on top of each other. These vertical arrangements help to identify quickly the subsequences in the sequence. By first inspection of the graph some subsequences become immediately visible. For example the vertical arrangement of crosses in the upper left corner (delineated by the solid blue line) suggests that a number of traces are very close to each. Detecting other structures in the graph is however not as straight forward as the first example. In the lower part of the graph (delineated by the blue dashed line) subsequences are less clearly visible. There is a "cloudy display" of data points that does not allow easy identification of subsequences. The cloudy display is the effect of the arbitrary ordering of the semantics on the vertical axis. An arbitrary or random ordering will set the semantics of traces that tend to appear together far apart from each other, making the visualization less effective.

The problem in hand relates to the ordering of the positions of semantics on the vertical axis. The temporal aspect of the graph is well served with the current setting because the relative temporal proximity of traces is clearly visible (vertical alignment of traces that close to each other). To enhance the informativeness of the graphical representation of traces, the positioning of the semantics on the vertical axis of the graphs has to be ordered. The ordering of the semantics on the graph has to allow semantics that are usually members of the same subsequence to be positioned close to each other on the vertical axis. Since the membership of the traces in subsequences is not known in advance the ordering can only happen by using the observations as these are made in the sequence.

### 3.3.2 Optimizing the ordering of dimensions for effective visualization

The interest in this thesis is to control the ordering of the semantics on a single axis (vertical axis), the optimization problem reduces to a single dimension, hence a unidimensional scaling problem.

The scaling problem consists of two parts:

- The distance matrix, which contains the item to item (dis)similarities. The distance matrix has to meet the metric properties [Gow86] so that the item to item relative dissimilarities can be retained after the scaling.
- A cost function that has to be minimized by an optimization algorithm in order to find the best positions for the semantics

The focus here is on the definition of a dissimilarity matrix that fits the information found in sequences of traces. For the cost function the least-squares criterion is used [Hub06]. To perform the optimization process, unidimensional scaling is used with the least square criterion $L_2$. The method is explained in short in 3.3.5.1. For further reading the reader is referred to [Hub06].

### 3.3.3 Association between traces

To measure the strength of the association between two traces an appropriate association measure is needed. Several association coefficients are found in the literature across different research domains [War08] [Che69]. The coefficients base their measurements on the observations that are made on how events occur in relation to each other, either in the time of in the space domain. To illustrate how observations

on events are made, an example is given. There are two types of events $A$ and $B$. The occurrence of the events is monitored on a daily basis (time domain). Four event counters are used to measure the phenomenon:

$x_{AB}$: *Event A and B occur*
$x_{AB}$: *Event A occurs but B does not* **(3-3)**
$x_{AB}$: *Event A does not occur but B does*
$x_{AB}$: *Event A does not occur and B does not occur*

At the start all counters are set to zero. Depending on what observations are made at the end of each day the corresponding counter is increased by one. To evaluate the association coefficient the counters are combined appropriately to convey the indented measure of association. Among the several coefficients that are found in the literature for this thesis the Jaccard is chosen. The Jaccard coefficient measures the ratio of common occurrences over the sum of all occurrences of the two events. The coefficient is chosen because it shifts the weight of measuring the association on the co-occurrences of events rather the common absences. The Jaccard coefficient ignores in its calculation the number of occasions where both events are absent i.e. $x_{\bar{A}\bar{B}}$ making the measure of association independent from the occurrence of other events. In the context of traces, the observation of two traces being both absent in a subsequence does not have an added value because it is not a statement on the functional coupling between two components. The Jaccard coefficient $\varphi$ has all properties of a metric [Gow86]. The coefficient $\varphi$ is defined as follows:

$$\boldsymbol{\varphi = \frac{x_{AB}}{x_{AB}+x_{AB}+x_{AB}}} \qquad \textbf{(3-4)}$$

### 3.3.4 Dissimilarity matrix for traces

The association of traces in a sequence is measured by their co-occurrence in subsequences. At this point however subsequences are not known. The sequence consists of ordered traces. To measure the association between any two traces in a sequence, the sequence is partitioned into a finite number of time frames. If a pair of traces falls within the same time frame, this is counted as a co-occurrence. This method provides an approximation of the association of traces since it is based on the occurrence of traces in these frames and not in the subsequences. The method is defined formally here.

A sequence of traces of length $T$ is partitioned into $n$ non-overlapping time frames $tf$ of width $l$, where , $l, n \in \mathbb{Z}$, so that

$$\boldsymbol{T = n \times l + r,} \qquad \textbf{(3-5)}$$

where $r, 0 \leq r < l$ is the remainder of the sequence. The total number of time frames after partition is $n' = n+1$ for $r > 0$, and $n' = n$ for $r = 0$. For every $tf_j$, where $j = 1, 2, 3, ..., n'$ the number of occurrences $c_{ij}$ are counted for each $ic_i$ that is found in the sequence, where $i = \{1, 2, ..., m\}$. Since the interest lies in the co-occurrence of traces and not in the number of occurrences, the count of the number of occurrences for a type of semantic $ic_i$ is transformed into a state element $b_{ij}$ obtained via a binary function:

$$b_{ij} = \begin{cases} 1 \ for \ c_{ij} > 0 \ (presence) \\ 0 \ for \ c_{ij} = 0 \ (absence) \end{cases} \quad \textbf{(3-6)}$$

The event counters (3.3-1) are updated for all pairs $< ic_i, ic_k >$, where $i,k = \{1,2,3,...m\}$. Based on these the Jaccard coefficient $S_{ik}$ is evaluated for all pairs $< ic_i, ic_k >$ of semantics. The association coefficients of all pairs form a similarity matrix $Sdt$ of size $m \times m$ that describes the pairwise associations of traces in the sequence. The dissimilarity matrix $Pdt$ that is required for the cost function of the optimization operation is acquired by $Pdt = 1 - Sdt$.

## 3.3.5 Optimization of the ordering using temporal associations

### 3.3.5.1 Optimization Method

The method that is used in this thesis is known as *random restarts unidimensional scaling* [Hub06] and will be described briefly here. Given an initial (random) ordering of positions, the optimization of the criterion $L_2$ (defined below) is approached through simple pair-wise interchange/rearrangement heurists. Such local interchanges continue until no further improvement in the criterion can be made. The minimum reached by this tactic is a local minimum and therefore not the optimal solution. To obtain a near to optimal solution, several local minima are obtained and the best out the set is chosen. To ensure a good approximation of the optimal solution, the set of local minima has to cover as much as possible of the input space. To achieve that, the process is initiated numerous times using random orderings $Y_0$ of the initial positions of traces as the input. The result is a set of local optima $L_{2\min}$ over the input space. The lowest value $\min\{L_{2\min}\}$ obtained from the *set* of local minima gives the approximated optimal solution for that run.

The least square criterion is defined as

$$L_2 = \sum_{i<k} (p_{ik} - |\, y_k - y_i \,|)^2 ,$$

where the element $p_{ik}$ in $Pdt$ is defined as $p_{ik} = 1 - s_{ik}$ and $y_i, y_k$ the set of coordinates (positions) of data type $ic_i$ and $ic_k$ respectively .

### 3.3.5.2 Effective ordering of positions of traces on the graph

To set the width $l$, domain knowledge is used. According to domain experts and the observations that were made during the fault injection experiments (3.2), the duration of a subsequence is not expected to exceed 300 seconds. The dissimilarity matrix $Pdt$ of the data types found in the data sequence is computed using the method described in 3.3.4. The *random restarts unidimensional scaling* method is used with the computed $Pdt$ as the input. The $\min\{L_{2\min}\}$ found suggests the optimal ordering $Y_{\min}$.

When the ordering $Y_{\min}$ that is obtained from the optimization method is used to plot the semantics on the vertical axis, the result is an improved visualization. This can be

seen in Figure 3-11. The sequence in Figure 3-11 is the same sequence as in Figure 3-10.

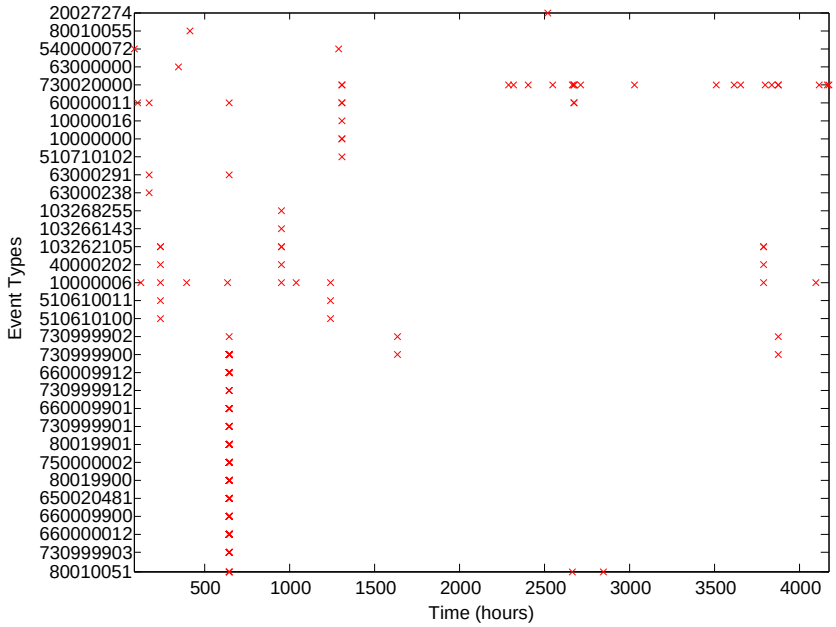



**Figure 3-11: Optimally reordered dimensions (data types) in y-x plot**

Subsequences become clearly visible when the ordering of the positions on the vertical axis is done based on the pairwise associations of traces. Visual exploration of the data sequences is made easier.

### 3.3.6 Case study: Visual exploration of temporal data sequences obtained from professional system operating in the field

This case study shows how the visual exploration of the sequence of traces, using the effective visualization technique described here, can help increase the knowledge over the formation of subsequences. In the context of this research, visual representation can help enhance or weaken the following *beliefs*:

1. Subsequences are dense formations i.e. traces located relatively close to each other compared.
2. Subsequences with similar structure (temporal and semantic) can be found in long temporal data sequences. The operating conditions in the field can have an effect on the structure of subsequences. Does this effect allow the formation of subsequences that show similarities the same way as the results from the fault injection experimentation?
3. Recovery subsequences follow error subsequences. The association between error and recovery subsequences is interesting because it is a data manifestation that is in line with the motivation of this thesis. The motivation of this thesis is to use information on the co-occurrences of error and recovery events to improve the availability of the system. The association between recovery and error subsequences was observed under the controlled conditions of the experimentation. How does the phenomenon occur in operational conditions?

4. Search for new unknown data structures. The data structures that are known are the subsequences. These are dense formations with a relatively short duration. Are there other data structures in the sequence that do not follow the same pattern?

Each of the 137 sequence in the sample set is represented visually and observations that are relevant to the three beliefs are recorded. The findings are presented in the following sections. Findings in relation of belief 1 and 2 are described in section 3.3.6.1 , about belief 3 in section 3.3.6.2 and about belief 4 in section 3.3.6.3.

### 3.3.6.1 Temporal structure and semantic content of subsequences

Using the visualization technique described here and by producing the graphical representations of all sequences in the data sample, it is observed that:

- Subsequences, error or recovery, are indeed found in the form of bursts. Example of this can be seen in (Figure 3-12 and Figure 3-13) where vertically aligned data points, i.e. subsequences, are formations of closely located traces and subsequent subsequences are separated by relatively long intervals. This observation enhances the confidence regarding *belief 1* and develops into *conjecture* 1 on the *temporal structure* of subsequences:

  *Traces of the same subsequence are found in close temporal proximity. Traces of different subsequences are separated by relatively long intervals of time.*

- Subsequences can re-occur consisting of the exact same traces (subsequences outlined with dotted line in Figure 3-12 and Figure 3-13). Also, subsequences can re-occur consisting partly of the same traces and partly of traces that are unique to each subsequence (subsequences outlined with dashed line in Figure 3-12, the traces shared among all four subsequences are colored in blue). This observation enhances *belief 2* and develops into *conjecture 2* on the *semantic content* of subsequences:

  *Semantics can reoccur together with a varying level of consistency to form subsequences.*

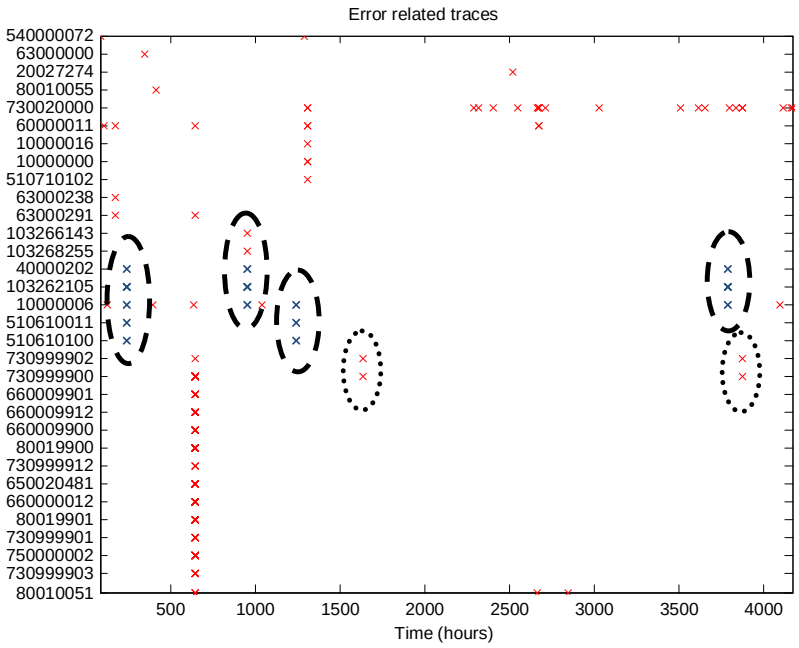



**Figure 3-12 Error subsequences**

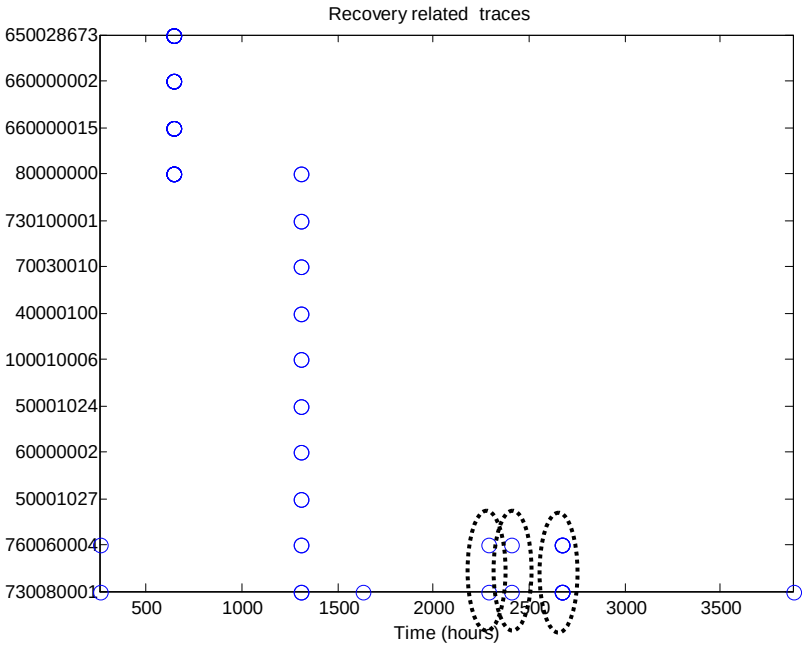



**Figure 3-13 Recovery subsequences**

### 3.3.6.2 Association between subsequences representing faults and recoveries

The association between error and recovery subsequences was first observed during the fault injection experiments where system failures where succeeded by automatic recoveries. The experiments were conducted in a controlled environment with specific operating conditions. Traces obtained from systems operating in the field can provide a new insight into this aspect because the information they contain represents operational conditions.

The visual exploration of the sampled data set provided strong evidence that such associations exist. Examples of such association can be seen in Figure 3-14 (Error

traces, are represented with red crosses, recovery traces are represented by blue circles).

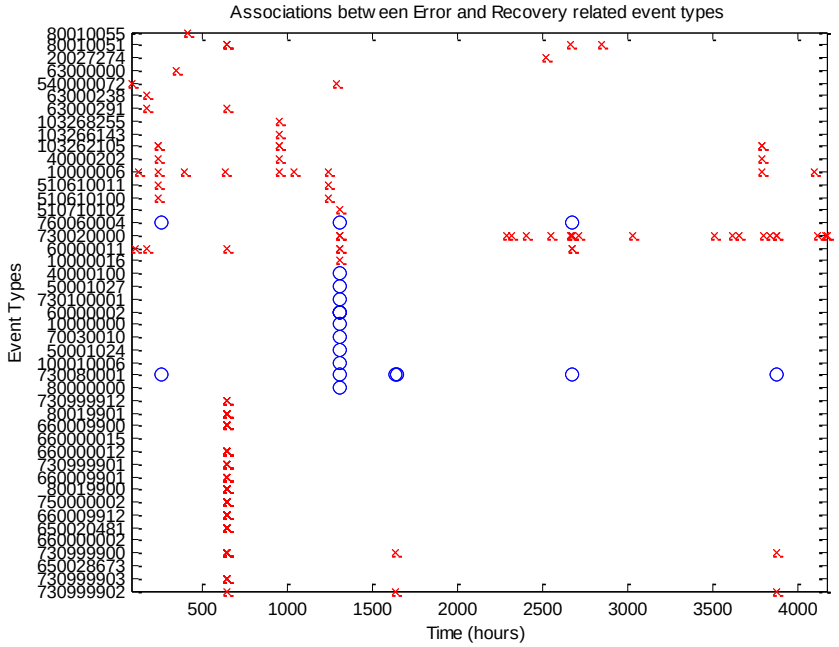



**Figure 3-14 Associations between error and recovery subsequences**

Associations between error subsequences and recovery subsequences can be seen where vertical alignments of crosses and circles exist. In the sample data, it was observed that recovery subsequences are preceded by error subsequences, which is enhancing the initial *belief 3* and leading to *conjecture 3* that:

*Subsequences representing recoveries follow subsequences representing failures.*

### 3.3.6.3 Discovery of unknown formations in long sequences of traces

Visual exploration helps to identify aspects of the data sequence that were previously unknown. Plots of sequences from the same sample set revealed the presence data structures that were previously not known. These structures seem to cover the entire length of the sequence with frequent entries that appear to be equally spaced. The structures are found in error and recovery traces (in Figure 3-15 and Figure 3-16) as data points that form horizontal lines.

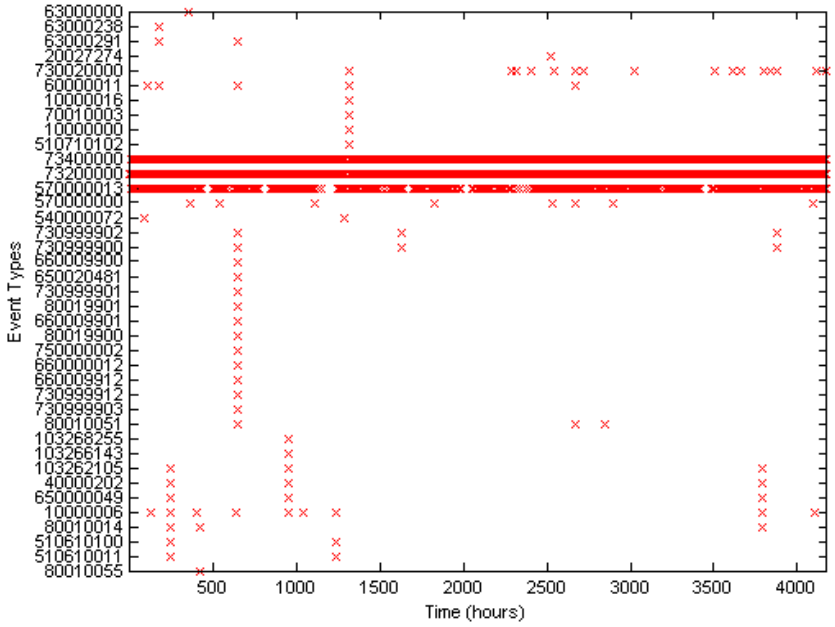



**Figure 3-15 Error traces forming dense horizontal structure**

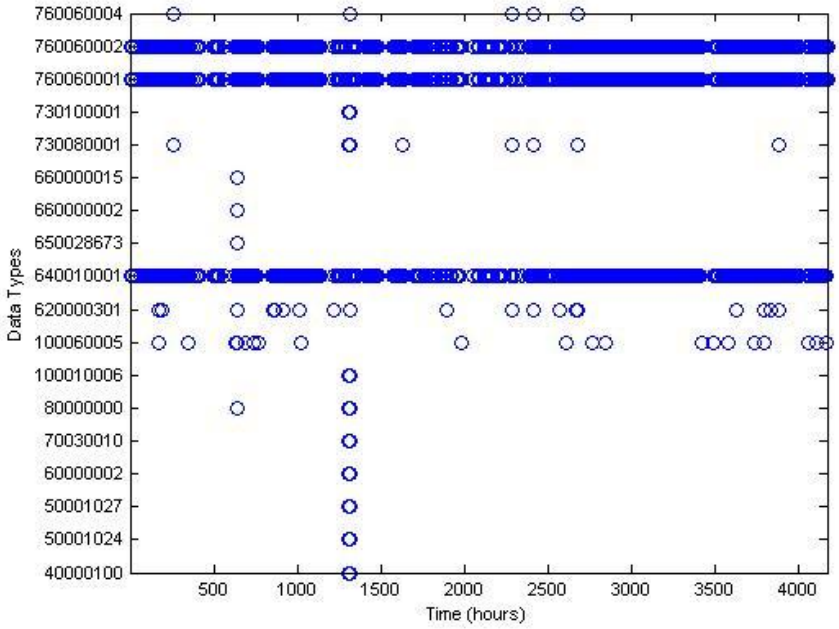



**Figure 3-16 Recovery traces forming dense horizontal structures**

This evidence supports *belief 4* and leads to the *conjecture 4*:

*Data structures that differ from the conventional structure of subsequences are present in the sequences of traces*

These data structures are identified as *partially periodic subsequences* (*pps*). Pps do not represent physical events that in similar manner as subsequences do. Pps are artifacts of the logging mechanism. Pps will be discussed in detail in section 3.4. As it will be shown in 3.4, pps are obstructing the transformation process and therefore

need to be removed to allow further processing of the sequences. In section 3.4, a technique for fast detection of pps in long sequences is presented.

### 3.3.7 Discussion and conclusions

Visual representation is a powerful tool in the exploration of data sets when little information is known about how the data is structured. Improving the graphical representation by optimizing the position of trace on the vertical axis can increase the power of visual exploration. The proposed method is an inexpensive ordering technique that can improve the effectiveness of the visual representation of traces. Its inexpensiveness lies in the fact that without deep prior knowledge on the formation of subsequences, the association of semantics can be measured quickly and effectively, by using a simple partitioning technique of the sequence with the combination of the appropriate measure of association i.e. the Jaccard coefficient. This measure, together with the use of a simple optimization algorithm can improve considerably the graphical representation of traces.

Exploring the sequences for structures without the effective representation would be virtually impossible. The exploration of sequences of the sample helped to increase the understanding on the structure of subsequences by getting a visual feel of the data structures that are present. Unknown aspects of the traces were discovered. The presence of *pps* is problematic for the implementation of data mining tools and need to be removed. Although *pps* can be removed by simply filtering out the traces that form them, their automated detection is not straight forward.

## 3.4 Detection of partially periodic subsequences

An important part of the preprocessing of traces is to remove any unwanted data structures that can hinder the transformation process. The visual examination of the graphical representations of the traces revealed such structures in the data sequences.
By closer examination it is found that that some traces form "horizontal" structures. Within these structures instances of the same trace *ic* are spread over long intervals. The instances are spaced apart in almost constant distances. This spacing continues for a period of time, intermits for some other period of time, only to resume at some point later following a similar pattern.

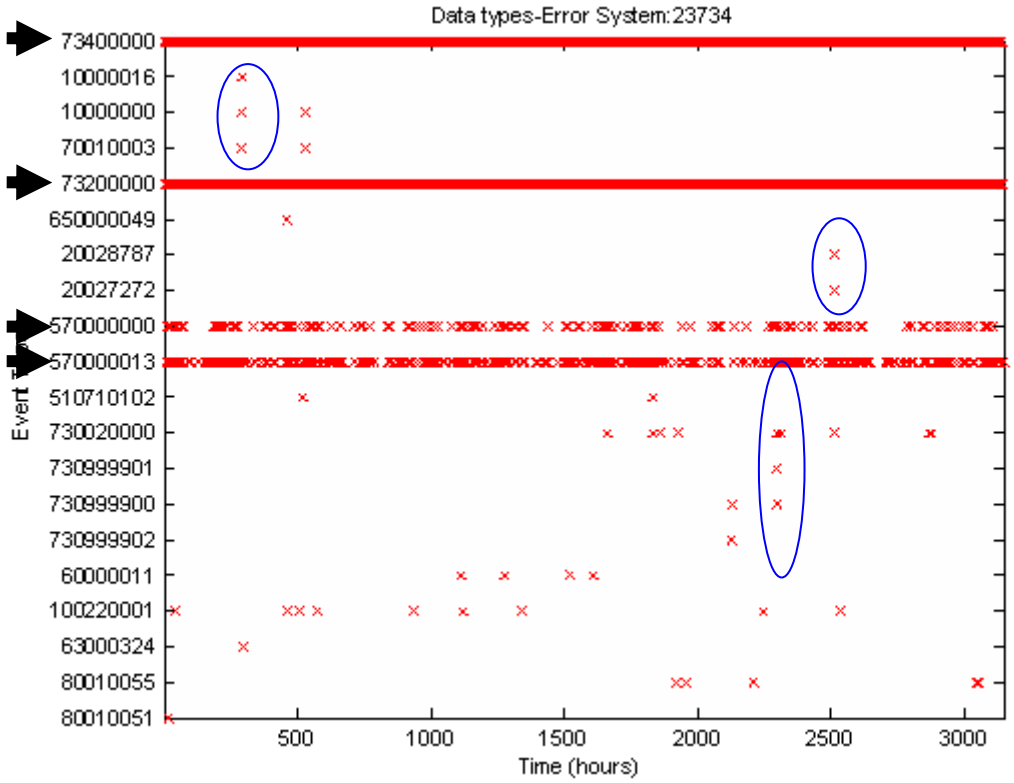



**Figure 3-17 Partially periodic subsequence in the audit trail**

Such data structures are known as *partially periodic patterns* or *partially periodic subsequences* (*pps*) [Ma01]. The term *periodic* refers to the constant interval between successive traces and the term *partially* to the fact the periodic structure is intermitting.

Pps are not the result of randomly occurring failures as the traces that appear in vertical structures, but are the product a semi-deterministic processes. They are likely to be the result of periodic monitoring mechanisms. These patterns occur for example when, due to an erroneous component, a periodic monitoring and recording mechanism is initiated. The purpose of such loggings is to give an alert to operators and make the problem visible to them when traces are visually inspected. Once the problem is solved the monitoring mechanism stops the recording of the data and the data structure terminates.

The presence of pps is problematic because as a data structure it masks the failure and recovery "vertical" subsequences to the discovery algorithms. Subsequences in the long sequence (encircled in blue in Figure 3-17) can be identified because of the compact arrangement of traces within the subsequences and the relatively long distances between subsequences. This arrangement is used by data mining techniques (chapter 4) for detecting subsequences in long sequences. The pps extends over long lengths in the sequence and masks subsequences by "bridging" the distances between consecutive bursts. As a result, subsequences of interest that otherwise would be clearly separated by relatively long periods of time, in the presence of pps become virtually invisible to data processing algorithms.

Because pps do not represent random physical failure events and its presence obstruct the further transformation of traces they are unwanted structures and should be either removed entirely from the sequence or replaced by another representation. Detecting the traces that exhibit pps structure is a large set of sequences, is a rigorous manual task. An automated method is needed that can search and find traces that follow pps patterns. The method has to examine recursively all types of traces in each sequence and detect these that exhibit pps structure. One of the challenges in detecting such patterns is that the period i.e. the length of the constant interval between subsequent traces in an on-segment, is not known in advance. Another problem is that, though the interval of the period is mostly constant it is not in an absolute manner. Small variations within the patter can exist.

A method is presented for detecting efficiently the type of traces that exhibit pps structure in a sequence. In section 3.4.1 the pps is described and defined formally. Section 3.4.2 presents current methods found in literature for detecting pps and discusses their shortcomings. In 3.4.3 am efficient method is presented for detecting pps in sequences of traces. Section 3.4.4 described a comparative study between the proposed method and the state of the art method found in literature. In section 3.3.5 a cases study is presented where the proposed method is applied on the sample sequences and the findings are discussed. Section 3.4.5 is concludes the discussion on pps.

### 3.4.1 Partially periodic subsequences

In contrast to a continuous periodic pattern where events reoccur continuously in regular intervals, *pps* periodic occurrence of events is present for a period in time, the on-segment, and ceases to exist for a following period phase in time, the off-segment. In Figure 3-18 two on-segments are shown where events are taking place at times $t_i$. In the pps purest form, successive events in on-segments are separated by intervals of constant length, known as the period p. The on- or off-segments can be of various lengths. In the example of Figure 3-18 the pps has a period of p=3 e.g. time units.

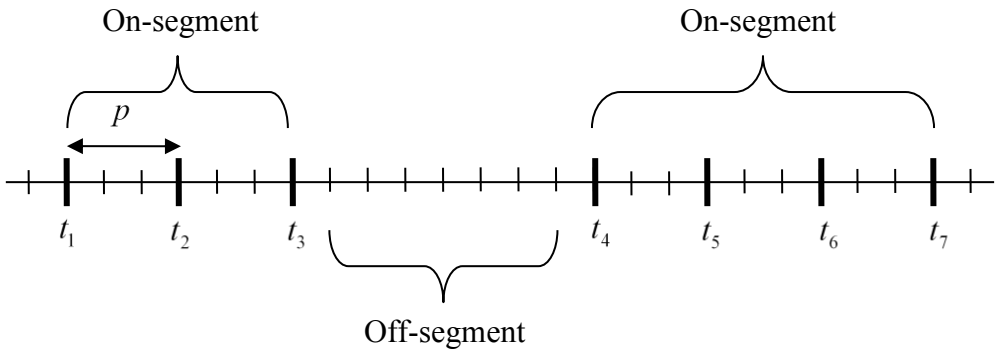



**Figure 3-18 Partially periodic subsequence (pps)**

In the context of the logging mechanism, the on-segment intervals correspond to the phases in time when a sensing mechanism is activated due to an error, in contrast to the off-segment intervals where the mechanism is inactive. If it is decided to replace the pps instead of removing it entirely, an option can be to replace each on-segment of the *pps* with a single trace of the same type. The single replacement trace should be located at the point in time where on-segment starts.

To detect a pps it has to be determined that a certain type of follows a pps pattern. The pps can appear with some degree of distortion in its structure. The distortion can be found in the form of additional traces of the same type placed randomly within the on-segment and adjacent to the traces that form the pure pps. Similarly additional traces can be placed in the off-segment closely to the border traces of the on-segments. In Figure 3-19 a *pps* with distortion is shown. The pps has two on-segments. The period is p=3 time units. The events that follow the period are indicated by the black vertical lines. Additional traces located randomly and adjacent to the segments of the pps are indicated by the red vertical lines.

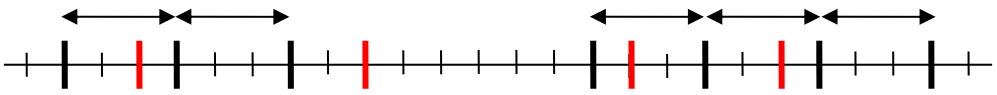


**Figure 3-19 Distortion in partially periodic subsequence**

Given that the period of a pps in not known in advance the presence of distortion can make the determination of the period p of the pps difficult because it blurs out the deterministic structure of the on-segments. The discovery of *pps* in the presence of such distortion is a challenge that requires specifically designed methods. Researchers have been working on the problem of *pps* detection.

### 3.4.2 Related work

The problem of discovering *pps* in data sets has been dealt by using different approaches. Ma et al. [Ma01], use association rules to discover *pps* in temporal data sequences. A different approach is used by Yang et al [Yan00]: a subsequence consists out of segments where the data type exhibits periodicity; the longest valid subsequence with periodicity is characterized as *pps*. A third approach is presented by Cao et al. [Cao07], where a symbol matching scheme is used in combination to shifting and comparing the sequence with itself to discover the *pps*.

Regardless of the approach, the biggest challenge for discovering which type of trace exhibits a pps structure in a sequence where multiple traces occur in different point in time, is that the period $p$ of the pps is not known beforehand. Some methods propose an exhaustive search of all possible lengths for period $p$, which is clearly not an efficient method if the amount of traces in the sequence is high. There is a need to narrow down the possible lengths for period $p$ in order to improve the efficiency of these algorithms. Such a method is proposed by Ma et al [Ma01], where binomial hypothesis testing is used. The test is designed to detect which interval length between successive events appears in greater than expected frequencies, under the assumption that all interval lengths are equally likely to occur. To perform the test, the difference between consecutive data points in the sequence is computed. The count of each interval length is compared against the 95% confidence level of the expected count for that interval length, under the assumption of equally likely occurrences. If the count exceeds that level, the interval becomes a candidate period i.e. an interval length that could potentially be the period of the pps. Once the set of candidate periods is defined, it is used to search for *pps* with one of the methods described above. In essence the objective of detecting candidate periods is to narrow down the number of trials for detecting the pps.

In a data sequence $S$ that consists of $N$ data points (events of the same type), $t_i$ represents the time of occurrence of a data point, where $i = 1, 2, ..., N$ and $t_i \neq t_j$ for $i \neq j$. The $n^{th}$ order difference is given by:

$$diff_n = t_i - t_{i-n}, \text{ for } \mathbf{1} \leq \boldsymbol{i} \leq \boldsymbol{N} \text{ and } \mathbf{1} \leq \boldsymbol{n} \leq \boldsymbol{N-1} \qquad \textbf{(3-7)}$$

and the 1st order difference is the difference between consecutive data points in $S$ and is given by $diff_1 = t_i - t_{i-1}$.

Identifying candidate period by using the 1st order difference comes with limitations. In the presence of distortion the 1st differences may not be the appropriate measure to detect the period of the pps. In Figure 3-20, an example of pps with distortion (in red) is shown.

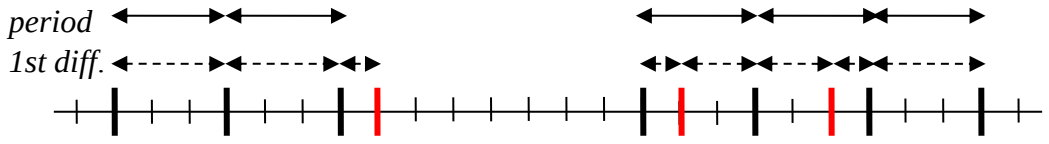



**Figure 3-20 Effect of distortion on period detection**

The intervals that indicate the period are represented by bidirectional arrows with solid lines and the 1st order differences by bidirectional arrows with dashed lines. In the example it can be seen how in the presence of distortion the 1st order difference misses to measure the correct number of intervals lengths that are equal to the length of the period. In the on-segment on the right, the first difference returns only one interval length that is equal to length of the actual period. This shortcoming was recognized by Ma et al [Ma01], and in their paper they suggest that for highly distorted data sequences, additional, higher order differences (e.g. the second, third etc.) can be used to perform the test. This is however not an efficient method since the

test has to be repeated several times, one for every order of difference. Also, the level of distortion is not known in advance and therefore it is not possible to foresee whether the second or third or other order of differences should be used.

Here a test is proposed that can detect correctly the presence of periods in a data sequence with high rates of success even with high levels of distortion, using a single round of computation. The test is built on the same principle of the binomial hypothesis test of Ma et al [Ma01], but instead of taking only a single order of differences at a time to make the comparison between counted and expected interval lengths, *all orders of differences* between the traces are taken at the same time.

## 3.4.3 Mixed Erlang test for detecting candidate periods in temporal data sequences

### 3.4.3.1 Mixed Erlang Distribution

For a data sequence $S$ that consists of $N$ data points, the differences between data points of all orders can be computed using 3.4-1. The result of the computations is represented in Table 3-3. The head row indicates the order of difference. For example the column of "1st diff" indicates all differences $diff_n = t_i - t_{i-n}$ of the order $n = 1$, the column "2nd diff" of the order $n = 2$ and so on. Each column contains the number of differences $D_n$ obtained when performing 3.4-1 on all data points in $S$ for a specific order $n$. For example the first column contains $D_1 = N - 1$ differences between all successive points in $S$. The number of differences reduces as the order $n$ increases.

| | **1st diff** | **2nd diff.** | **3rd diff.** | | **$n^{th}$ diff** | | **$(N-1)^{th}$ diff** |
|---|---|---|---|---|---|---|---|
| | $t_2 - t_1$ | | | | | | |
| | $t_3 - t_2$ | $t_3 - t_1$ | | | | | |
| | $\cdots$ | $\cdots$ | $\cdots$ | | | | |
| | $t_i - t_{i-1}$ | $t_i - t_{i-2}$ | $t_i - t_{i-3}$ | $\cdots$ | $t_i - t_1$ | | |
| | $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | | |
| | $t_N - t_{N-1}$ | $t_N - t_{N-2}$ | $t_N - t_{N-3}$ | $\cdots$ | $t_N - t_{N-n}$ | $\cdots$ | $t_N - t_1$ |
| number of differences | $N-1$ | $N-2$ | $N-3$ | $\cdots$ | $N-n$ | | 1 |

**Table 3-3 All Differences of all orders between all data points in $S$**

The number of differences obtained from $S$ for the $n^{th}$ order is $D_n = N - n$, whereas the total number of differences obtained from the 1st until the $n^{th}$ order is given by:

$$K = \sum_{n=1}^{N-1} (N - n) = \frac{1}{2} N(N-1) \qquad \textbf{(3-8)}$$

Assuming that is $R$ a random data sequence, the data points $t_i$ for $i = 1, 2, ..., N$ follow a Poisson process with parameter $\lambda$. By the properties of the Poisson process, the set of differences $diff_n$ of the $n^{th}$ order is a sample from an Erlang $E(n, \lambda)$ distribution with

density [Cox62]. Let $X$ be a continuous random variable that represents the set of differences $diff_n$ of $n^{th}$ order with density:

$$f(x \mid \lambda) = \frac{\lambda^n x^{n-1} e^{-\lambda x}}{(n-1)!} \qquad \textbf{(3-9)}$$

If the samples of the differences of all orders *n*, where $n = \{1, 2, 3, ..., N-1\}$ are put together the distribution becomes a mixed Erlang, where each $E(n, \lambda)$ has a mixing proportions $q_{N.n}$

$$q_{N,n} = \frac{N-n}{K} \qquad \textbf{(3-10)}$$

The mixed density is thus

$$f(x) = \sum_{n=1}^{N-1} q_{N,n} \times \frac{\lambda^n x^{n-1} e^{\lambda x}}{(n-1)!} = \sum_{n=1}^{N-1} \frac{2(N-n)}{N(N-1)} \times \frac{\lambda^n x^{n-1} e^{\lambda x}}{(n-1)!} \qquad \textbf{(3-11)}$$

with first moment

$$\mu_N = \frac{1}{\lambda} \sum_{n=1}^{N-1} n q_{N,n} \qquad \textbf{(3-12)}$$

#### 3.4.3.2 Binomial hypothesis test for nth order differences

To detect candidate periods $p_c$ a binomial hypothesis test is constructed. Given a sequence S, the test compares the observations $O_{l_i}$ (counts) made of an interval length $l_i$, where $i = \{1, 2, 3, 4, ...\}$ computed from the differences of all orders *n*( $n = \{1, 2, 3, ..., N-1\}$), with the expected number of intervals $E_{l_i}$ assuming a random sequence. In specific the null hypothesis is formulated as follows:

$H_0$: The count of intervals of length $l_i$ is obtained from the differences of all orders from a random data sequence.

And the alternative hypothesis states:

$H_1$: The count of intervals of length $l_i$ is obtained from the differences of all orders from a non-random data sequence.

Non-random in the context of traces can be either a pps or an entirely periodic data structure (on-segment only).

To test the hypothesis the observations $O_{l_i}$ made for each interval length $l_i$ is compared against the expected number $E_{l_i}$. The expected number follows a binomial distribution $B(K, P_{l_i})$, where *K* is the total number of intervals of any length and $P_{l_i}$ the probability of an interval of length $l_i$ occurring.

To test the hypothesis the standardized deviation from the expectation $E_{l_i}$ is used:

$$Y_i^2 = \frac{(O_{l_i} - E_{l_i})^2}{E_{l_i}(1 - P_{l_i})} \qquad \textbf{( 3-13)}$$

The statistic derives from the sampling distribution of the variance [Lan69]:

$$\sum_{z=1}^{z=r} Y_i^2 = \sum_{z=1}^{z=r} \left( \frac{X_i - \mu}{\sigma} \right)^2 \sim \chi_r^2 \qquad \textbf{( 3-14)}$$

in particular:

$$Y_i^2 \sim \chi_1^2 \qquad \textbf{( 3-15)}$$

and is used on the basis of the central limit theorem that states that for sufficiently large samples ($\geq 30$) the sampling distribution of the variance of a variable $X_i$ obtained from a non-normal distribution approaches a chi-squared distribution ( $\frac{Xi - \mu}{\sigma}$ approaches the standard normal).

The statistic $Y_i^2$ uses the true mean $\mu = E_i = K \times P_i$, where $E_{l_i}$ the expected number of intervals of length $l_i$, $K$ is the total number of intervals (3.4-2) and $P_i$ the probability of an interval of length $l_i$ and the true variance $\sigma^2 = E_{l_i}(1 - P_{l_i})$.

The probability $P_{l_i}$ is computed by condensing the probabilities of the cumulative mixed Erlang distribution:

$$P_i = P\ l_i - d \leq x \leq l_i + d\ = \int_{l_i - d}^{l_i + d} f_x\ x\ dx \qquad (3\text{-}16)$$

where *d* is an arbitrary real number.

When $Y_i^2$ exceeds the value of $\chi^2 = 3.84$ (for a 95% confidence level), the null hypothesis can be rejected, the observations $O_{l_i}$ for interval length $l_i$ cannot originate from a random data sequence and the interval length $l_i$ becomes part of the set of candidate periods $p_{cnd}$.

#### 3.4.3.3 Reducing the number of candidate intervals

The hypothesis test can return a set of candidate periods $p_{cnd}$ containing more than one interval length. Within this set of candidates it is likely that there are interval lengths that are multiples of other lengths. That happens because a higher order difference can produce interval lengths that are multiples of intervals from a lower order difference.

For example an interval of length $l_c$ is computed taking the 1st order difference between traces. If $l_c$ is a candidate interval i.e. its counts exceed the expectation according to the mixed Erlang, the 2nd order difference will produce intervals of length $l_{c'} = 2l_c$. Also the count of $l_{c'}$ is likely to exceed the expectation and therefore become a candidate period. The count of $l_{c'}$ is an artifact of the computation method. Only the interval $l_c$ should become a candidate period. To prevent such artifacts,

interval lengths in set of candidate periods that are multiples of other interval lengths are excluded. The candidate periods are examined in ascending order of interval length. The shortest intervals lengths are kept whereas intervals that are multiples of those are excluded.

### 3.4.4 Assessing the effectiveness of proposed detection method

The effectiveness of proposed method for detecting candidate intervals in the presence of distortion is assessed with an experiment. A pps is produced with period $p = 5$ and total length $T = 500$ time units. The pps has three on-segments and two off-segments of various lengths. The method described in this section is used to find candidate periods. If the period $p=5$ is within the set of candidate periods the tests is successful, otherwise it that tests results to failure.

To simulate distortion, additional data points are added randomly to the pps. The level of distortion is increased gradually and the search for candidate period is repeated. To measure the level of distortion for each run, the distortion-to-signal ratio (DSR) is used. The ratio indicates the number of additional random data points added to the pps over the number of pps data points. The experiment starts with the DSR set to zero and gradually increased to 4.

The model proposed here based on the mixed Erlang (model A) is compared to the performance of the model suggested by Ma et al [Ma01] (model B). For each DSR level 100 runs are conducted. Each run can result to a success or a failure, depending whether the correct period ( $p = 5$ ) is detected or not. The average success rate (ASR) for all runs is indicating the performance of each model in the test. In addition, the number of false positives is counted on each distortion level i.e. intervals of lengths other than 5 are added to the set of candidate periods. The average number of false positives is computed for the 100 runs as well.

In Figure 3-21 the results of the analysis can be seen. The **solid line represents the results of model A** whereas the **dashed line model B**. Model A outperforms model B undoubtedly.

For the first runs of the test the results of model B agree with those reported by Ma et al. [Ma01]. The average success rate of model B remains high until the DSR reaches the value of one, where the number of distorted traces is equal to the number of signal traces. Then the ASR drops rapidly and becomes equal to zero when the DSR is equal to two. For model A the ASR keeps perfect score even when the DSR exceeds the value of 2. At this point there are on average 2 additional randomly positioned data point for each data point in the pps.

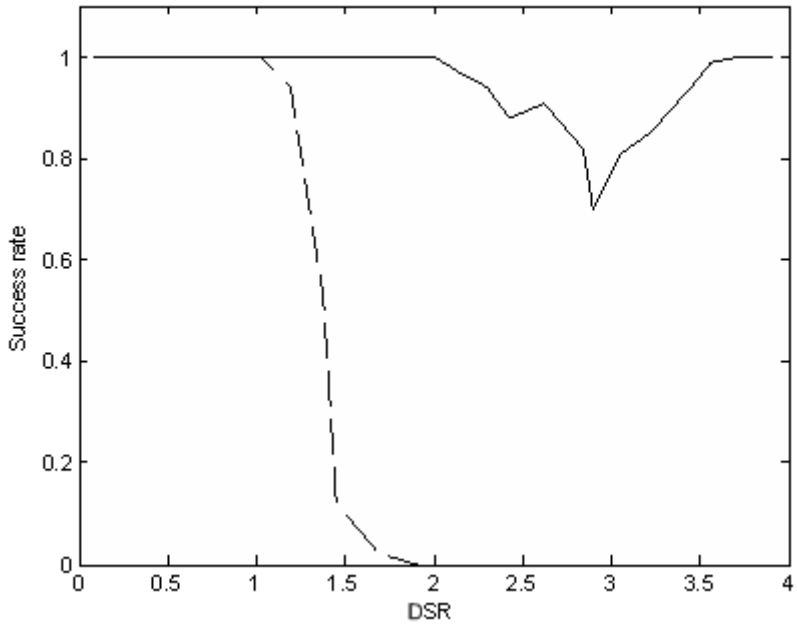



**Figure 3-21 Average success rate vs. DSR**

Another observation made is that the performance of model A seems to increase again once the DSR takes values $> 3$. This can be explained in the following way:

The pps of the experiment has a period of $p = 5$. Once the DSR approaches the value of 3, there will be on average 3 additional randomly positioned data points every two *pps* data points. This means that the interval of length $l_1 = 1$ becomes the shortest dominant interval and is recognized as a candidate period. Because the interval of length $l_5 = 5$ is a multiple of $l_1$, it is recognized *again* as a candidate period. Consequently the curve of ASR reverses from decreasing to increasing and reaches perfect score again when DSR becomes equal to 4. At this point there are some many randomly positioned data points that they are present at every time unit. The pps has been transformed from having p=5 to having p=1. In a real situation a data sequence similar to that of the experiment at the point where $DSR > 3$ will be correctly recognized as a *pps* with period $p = 1$.

The number of false positives provides additional information to the performance of the two methods. False positives reduce the efficiency of the mining algorithms by falsely increasing the number of candidate periods. In Figure 3-22 the average number of false positives (AFP) against DSR is shown. **The solid line represents the AFP for model A** and the **dashed line represents the AFP for model B**. In Figure 3-23 the difference of the numbers of false positives produced by the two models is shown ($Comp_{AB} = False\ positives\ of\ ModelA(DSR) - False\ positives\ of\ ModelB(DSR)$).

It can be seen how the two models perform identically ($Comp_{AB}$ in Figure 3-23) as long as the DSR level remains below 1. From the DSR level of 1 until the value of 2, model A produces increasingly more false positives, but still in low numbers. At this point there are 470 interval lengths that are put to test and the AFP is 20. Moreover at this point model B cannot identify the correct interval as candidate period. From this

point on the AFP of model A continues to increase with the same rate until the DSR takes values $>3$ where the period switches from $p=5$ to $p=1$.

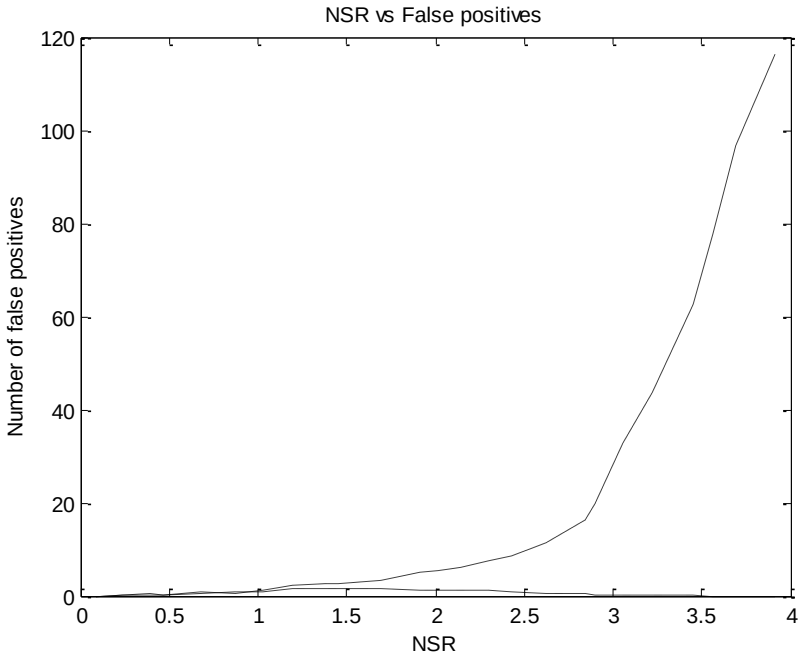



**Figure 3-22 Average number of false positives vs. NSR**

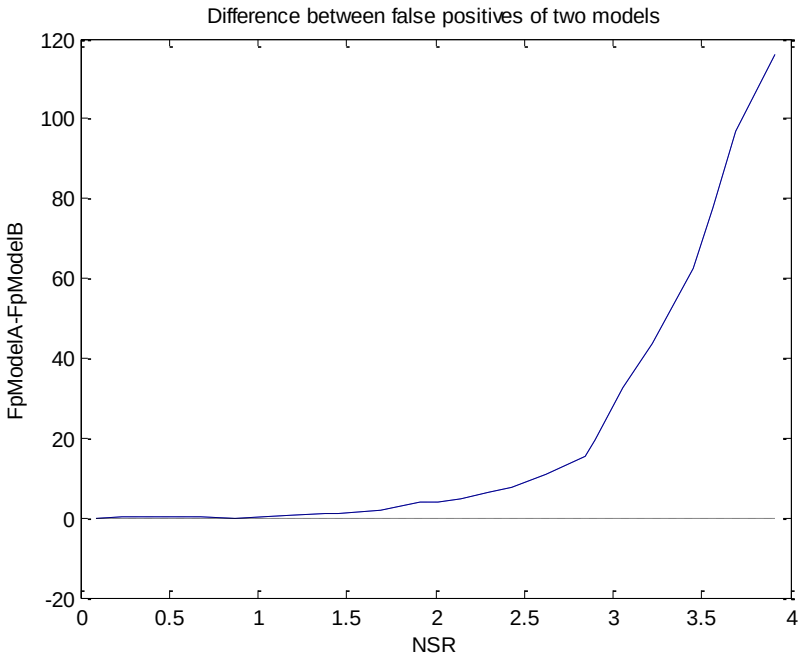



**Figure 3-23 Difference between numbers of false positives for two models**

### 3.4.5 Case study: pps in sequences of traces

Sequences from 137 systems operating in the field are prepared for the transformation. At this phase of the data preparation the objective is to identify the trace types that are *pps* end remove them from the temporal data sequence.

### 3.4.5.1 Identifying candidate intervals in system event loggings: **analysis**

The first step in identifying a *pps* is to detect any candidate intervals using the mixed Erlang test that was discussed in this section. For every sequence in the sample data set, each trace type is tested whether it has candidate intervals.

A sequence of traces $S$ contains $N$ entries. For each type of trace $ci_j$ (represented by its id), where $j = \{1, 2, 3, ..., k\}$, the test needs to be applied separately to determine whether it is a pps. To do that, for each type of trace $ci_j$, data sequence $S_j^{'}$ is produced from $S$ that contains only traces of a that trace type, by filtering out any instance of all other $k-1$ and keeping the temporal information (timestamps) intact. The resulting subsequence $S_j^{'}$ is of length $T' \leq T$ and contains $N_j^{'}$ traces, where $N_j' \leq N$.

In Figure 3-24 an example of a data sequence is shown containing error traces only. In this data sequence 20 different types of traces can be identified (vertical axis). For this example the test has to be performed 20 times, once for every type of trace.

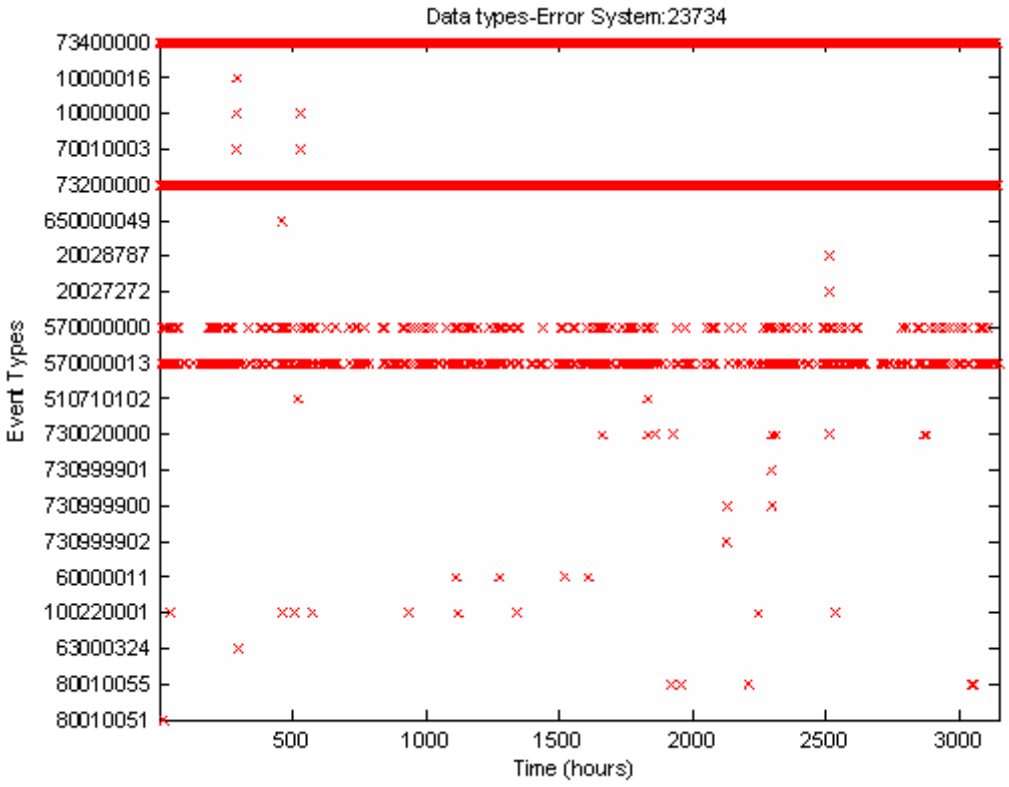



**Figure 3-24 20 different types of traces in a sequence**

The mixed Erlang distribution is used to establish the null hypothesis under the assumption of random temporal locations of traces in $S^{'}$. The null hypothesis allows to test whether the counts of interval lengths computed for $S^{'}$ are higher than what is expected under the assumption of randomness. To achieve that the counts in $O_{l_i}$ are

compared to the expectation $E_{l_i}$ for a mixed Erlang distribution $f_X(x)$, with parameters $N'$ and $\lambda_{ci_j} = \frac{N_j'}{T_j'}$.

Given that *pps* are the result of build-in monitoring mechanisms, traces that are found to be *pps* in one system could be *pps* also in another. This assumption can be extended to the period that defines the *pps* and consequently to the candidate intervals that were detected.

The procedure is executed with Matlab. Results are presented in the following subsection

### 3.4.5.2 Identifying candidate intervals in system event loggings: results

In all data sequences examined from 137 different systems, 202 different trace types were found. Out of these trace types, 8 were found to be a *pps* in the sequences of *all* systems in the sample. One type of trace has period of $p = 1\,\text{sec}$ and the other seven have $p = 898\,\text{sec}$. The results are consistent across all sequences in the sample i.e. if a trace type is found to form a *pps* in a sequence it is can be found only as *pps* in other sequences with the same period. All instances of trace types that are *pps* are removed leaving sequences with only burst like subsequences (Figure 3-25).

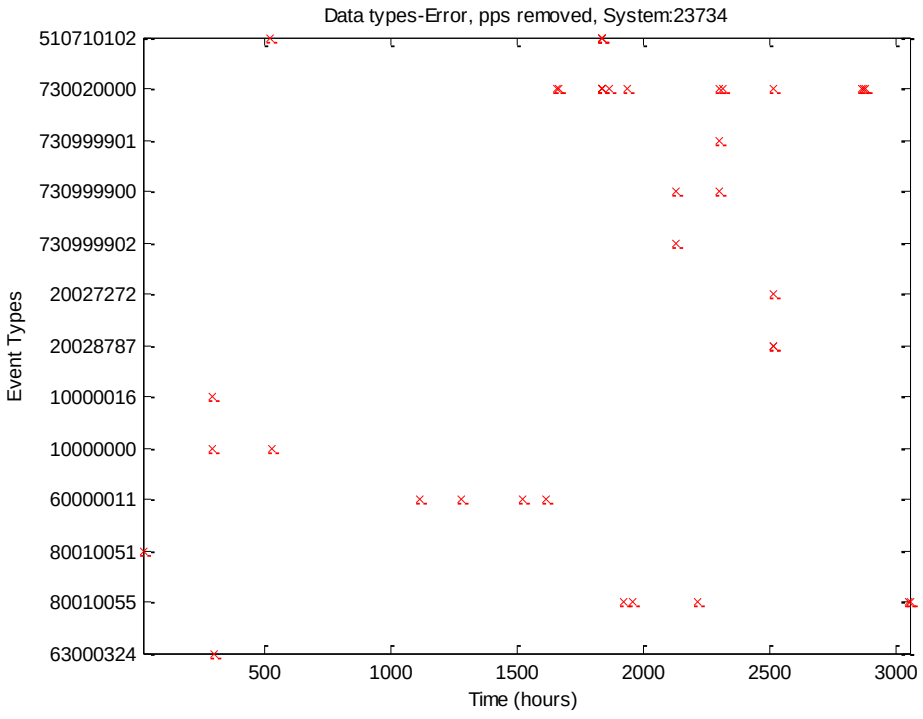



**Figure 3-25 Sequence cleared from *pps***

Removing the *pps* allows for further processing of the data sequences. In the next chapter a method for identifying subsequences in a data sequence will be presented. To achieve that, an unsupervised learning method is used.

### 3.4.6 Discussion and conclusions

In this section a method was presented that allows the identification of candidate intervals for detecting *pps*. A binomial hypothesis test is proposed to identify the candidate intervals. The test is based on comparing the occurring frequencies of interval lengths with the expected frequencies under the assumption of randomness in the locations of data points. The mixed Erlang distribution is used to estimate the expectations.

In the experiment the performance of the proposed model is high for increasingly high DSR levels outperforming an alternative method proposed by Ma et al [Ma01].

# Chapter 4

# 4 Detection of subsequences in sequences

Long sequences of traces can extend over periods of many operating hours and can contain thousands of entries. The traces that originate from single error or recovery events form subsequences. Subsequences are data structures that are characterized by the relatively close temporal proximity of its traces. Identifying the subsequences in a long sequence of traces is the first step of the transformation process and it is the step that defines the locations for the representations of physical events. Because the amount of traces in a sequence can be very high, the subsequence discovery process has to be performed automatically.

The method for subsequence discovery in sequences has to be system generic, capable of performing independently of the type of system or its use, using solely the information that can be found in the sequence. In this chapter a method for detecting subsequences in long sequences of traces is presented that is based on the structural characteristics of the subsequences i.e. the relatively close temporal proximity of traces within the same subsequence compared to the distant temporal proximity of traces between different subsequences (conjecture 1, section 3.3.6.1). The knowledge that allows the formulation of this structural characteristic of subsequences has been obtained by the exploratory analysis discussed in Chapter 3.

The process of detecting subsequences in a long sequence of traces is referred to as *segmentation*. An unsupervised clustering algorithm is used for the detection of subsequences in long sequences of traces. The term "unsupervised" suggests that there is no external validation data that can be used to train the algorithm. To guide the segmentation operation, a measure of cluster separation is used. The measure is used to choose the segmentation of the sequence that satisfies best conjecture 1.

This chapter is organized as follows: previous works found in literature, on the topic of segmentation of temporal data sequences are presented in 4.1. In 4.2 the proposed framework for segmenting long sequences of traces is presented. In subsections 4.2.1-4.2.5 each step of the framework is described in detail. The emphasis is put on the robustification of the segmentation method against variation in the subsequences, which is described in subsection 4.2.3. The chapter closes with the discussion and the conclusions on this stage of the transformation process in section 4.3. The segmentation method described in this chapter is applied in the case study of chapter 7.

## *4.1 Related work*

Different methods for detecting subsequences in long sequences are found in the literature. In their most dependent form these methods require *a priori* knowledge on the specific form of subsequences. The specifics of the form can be the number and the type of traces that define a subsequence, or even the order of occurrence of the traces. Based on this knowledge string matching techniques are used to search for the subsequences in the sequence that best match the already known forms [Ant01][Che98]. To obtain such information on the form of subsequences requires extensive failure injection to create a knowledge base of cause and symptom relationship. Fault injection is an expensive exercise, especially for complex

professional systems where the number of different forms can be high. Moreover such a process can never be exhaustive to cover all possible types of errors that can occur in a system operating in the field.

Sequential clustering techniques have been used for segmenting data sequences, by identify patterns using the semantic information of subsequences [Cao05][Man97][Das73]. When an adequate number of matching subsequences is found (defined by the user how many), the subsequence is regarded as a pattern. This approach is appropriate when subsequences come in great numbers. Enough evidence has to be collected in relatively short period of time to support the existence of patterns. The time required to collect enough support in order to define patterns depends on the failure rate and the variety of types of failures. Professional systems are in general reliable products and failure events are relatively rare.

The detection of subsequences in long sequences of traces can be based solely on the temporal structure of subsequences. Such an approach is proposed by Tsao [Tsa83]. The proposed method uses the distance between two successive traces as the *cutoff parameter* to decide whether consecutive traces belong to the same subsequence. However Tsao defines the value of the *cutoff parameter* arbitrarily. Though this method is suitable for segmenting a sequences based solely on the structure of subsequences, the arbitrary setting of the *cutoff parameter* is not satisfactory, since it does not allow the segmentation operation to adjust to the characteristics of different sequences e.g. differences in the compactness of subsequences. In this thesis Tsao's method is used as the basis to guide segmentation, but it is developed further to include a criterion that can help to decide on the best value for the cutoff parameter. The segmentation of a sequence comes also with some risks of erroneously assigning traces to certain subsequences. These risks are particularly relevant when the value of the cutoff criterion is decided based on the information collected from a sampled sequence, but the same value has to serve the uses of field applications where new subsequences will arise. Finally an unsupervised clustering method required some form of validation. These considerations are discussed in the following section. A framework is proposed that is addressing these considerations.

## 4.2 Unsupervised segmentation of a long sequence of traces

The closeness of the temporal proximities of traces in the sequence indicates whether they are "members" of the same subsequence or not. Based on this observation Tsao [Tsa83] defines subsequences by using the cutoff parameter i.e. a threshold distance between consecutive traces, to decide whether these belong to the same subsequence or not. The clustering operation is performed by a sequential clustering algorithm. Tsao sets the value of the cutoff parameter arbitrarily. Setting arbitrarily the value of the cutoff parameter can result to errors in the segmentation of subsequences. What is an adequate cutoff value for one subsequence can be too short for another, which will result to mistakenly splitting traces when they should belong together. The problem can occur also in the other direction where a cutoff value that is set too high results into merging together traces that actually belong to different subsequences. The above problems are known in the literature of temporal data sequences [Han92] as:

1. The risk of *truncation*: the risk of assigning traces to different subsequences that should belong to the same subsequence.

2. This risk of *collision*: the risk of grouping together traces into one subsequence that actually belong to two or more different subsequences.

Between the two types of risks the risk of truncation is the most relevant risk in the segmentation of sequences coming from professional systems.

It was shown in chapter 3 that subsequences can vary in their temporal structure i.e. the distance between consecutive traces in a subsequence. Although subsequences tend to consist of traces that are placed densely next to each other (compact structure), other subsequences can extend over longer periods with longer intervals between successive traces (loose structure). Variation can be also found in the structure of subsequences that result from different instances of the same type of physical event e.g. same type of failure. Different instances can have more compact or less compact structures[1]. Such differences can be attributed to the operation of the logging mechanism. The mechanism can fail to produce traces, or record multiple traces instead of one. Delays in logging can result to recordings of messages in later point in time than the exact time of error [Han92]. It has been also shown that the level and the type of the system workload have an effect on the way failures are represented by traces [Iye82]. The variation in subsequences can result to truncation during the segmentation of a new sequence if the cutoff criterion is set to a low value based on the observations made in sampled sequences with compact subsequences.

Professional systems tend to be reliable products. Their failure rates are low. The creation rate of subsequences follows that failure rate of the system. It is expected that sequences of professional systems are well separated. To avoid speculating on a representative failure rate of professional systems, it is adequate to say that the interval between consecutive subsequences is typically measured in hours of operating time. On the other hand it was shown in chapter 3 using the fault injection that the length of subsequences can extend over approximately 300 seconds. More importantly the distance between successive traces in a subsequence is of lengths of few seconds. This structure enables the identification of subsequences using the information on the relative temporal location of traces in the sequence.

Instead of setting the value of the cutoff parameter arbitrarily, an appropriate criterion is needed that will take into account the compactness of all subsequences in a sequence. In the literature of data mining and particularly under the area a data clustering, many criteria exist for different types of needs [The09]. For the sequence segmentation, the aim is to find compact clusters. A criterion that favors the discovery of compact clusters is the cluster separation measure (CSM).

The value of the cutoff parameter is set with the help of the CSM and a sample sequence. This value fits the overall characteristics of the sample sequence and returns the most compact clusters for that sample sequence. However when the algorithm is put in use, it operates on newly formed sequences that can vary from the subsequences found in the sample. This variation in the structure of subsequences can lead particularly to higher risks of truncation. To reduce the risk of truncation in applications, the value of the cutoff parameter needs to be increased to an adequate

[1] Even though the variation in the structure of subsequences can be found in both aspects of the structure: temporal and semantics, in this step of the transformation process the interest lies in the temporal structure.

level that can tolerate variation in the subsequences. Given that there is only one sampled sequence available, variation in the subsequences has to be simulated. The variation is simulated using a random data sampling method that can produce variants of the original data sequence [Lev01]. The random sampling method is referred to as the *resampling method*. The value of the cutoff parameter has to perform equally well on the original as on the variant sequences. To measure this performance a new criterion is defined, the *hybrid cluster separation measure hCSM.* The process of selecting a value for the cutoff parameter that performs well according to the hCSM is referred to as *robustification.* The hCSM is the primary criterion to decide which value of the cutoff parameter to use in field applications for the segmentation of a sequence.

The risk of collision is perceived as a low risk here. Nevertheless it is advised to choose a value of the cutoff criterion that reduces the risk of collision. The risk of collision is quantified by the collision probability (CP), which is estimated using a method proposed by Hensen [Han92]. For the segmentation process the CP is used as a secondary criterion.

Given that no external data are available to validate the segmentation results, an internal validation method is used to assess whether the segmentation is the result of the inherent data structure found in the sequence or whether the segmentation that is obtained is likely to be the result of a random data structure.

According to the above, the framework for the segmentation consists of five elements:

1. A sequential clustering algorithm to segment the sequence. For each value of the cutoff parameter the clustering algorithm produces a segmentation of the sequence (4.2.1)
2. The CSM to decide which value of the cutoff parameter return the most compact clusters. (4.2.2)
3. Robustification of the value of the cutoff criterion to account for structural variation of subsequences using the resampling method and the hCSM (4.2.3)
4. Chose a robustified value of the cutoff criterion that also reduces the risk of collision during the application (4.2.4)
5. An internal validation criterion to verify that the segmentation of the sampled sequence using the selected value of the cutoff parameter is not the result of randomly positioned data points being grouped together (4.2.5)

In the following sections each one of the elements of the segmentation framework is discussed in detail.

### 4.2.1 Segmentation of temporal event sequence using a sequential clustering method

Each trace in the sequence is represented by a data point $t_i$ (data points represent traces of all types), where $t_i$ is the time of occurrence. The sequence is represented as an ordered set of *N* data points (traces) $S = \langle (t_1), (t_2), ..., (t_N) \rangle$, where $t_1 < t_2 < ... < t_N$. A sequential clustering algorithm is used to segment the sequence. The clustering operation starts with the first data point in the data sequence $t_1$. The operation continues sequentially for every data point in the data sequence until all points have

been processed. Throughout the chapter the Euclidean distance is used as a measure of distance.

The clustering with the sequential algorithm operates as follows:
The first cluster $C_1$ is formed by the first data point $(t_1)$ found in the data sequence *S*. For the next data point $(t_2)$ in the sequence *S*, a decision is made, whether to assign the data point $(t_2)$ to the existing cluster $C_1$ or to form a new cluster $C_2$. The decision is made by comparing the distance $d(t_2, C_1)$, between the first cluster $C_1$ and the data point $(t_2)$, against the value $\theta$ of the *cutoff parameter* $\Theta$, where $\theta \in \mathbb{N}$ and $0 \le \theta \le t_N - t_1$. If $d(t_2, C_1) \le \theta$ is true, the data point $(t_2)$ is added to the current cluster $C_1$. If $d(t_2, C_1) > \theta$, a new cluster $C_2$ is formed that contains at this stage only the data point $(t_2)$. The next data point $(t_3)$ is examined in similar manner, by comparing its distance from the last formed cluster i.e. $d(t_3, C_2)$. The last formed cluster is referred to as the current cluster. The algorithm continues this operations until all *N* data points in *S* are processed.

The distance $d(t_i, C_m)$ between a data point $(t_i)$ and the current cluster $C_m = \{(t_{i-1}), (t_{i-2}), ..., (t_{i-l})\}$ is $d(t_i, C_m) = t_i - t_m$, where $t_m = \max\{t_{i-1}, t_{i-2}, ..., t_{i-l}\} = t_{i-1}$, therefore $d(t_i, C_m) = t_i - t_{i-1}$
(Figure 4-1 vertical lines represent data points in the sequence, the blue ellipse represents cluster $C_m$).

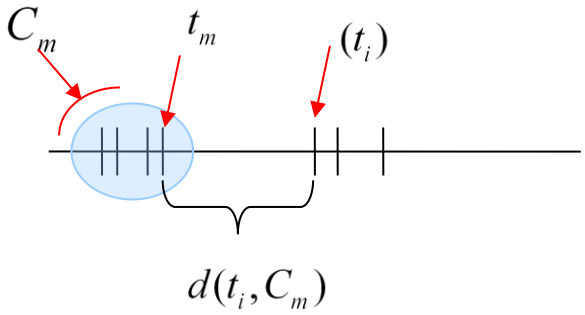



**Figure 4-1: Sequential clustering algorithm**

The pseudo code for this algorithm is given

- *m=1*
- $C_1 = \{t_1\}$
- *for i=2:N*
  - *if* $d(t_i, C_m) > \theta$ *then*
    - *m=m+1*
    - $C_m = \{t_i\}$
  - *else*
    - $C_m = C_m \cup \{t_i\}$
  - *end if*
- *end for*

For a data sequence containing $N$ data points a clustering result $C^R$ can contain $k$ clusters, where $1 \le k \le N$ depending on the value of $\theta$. For $\theta = 0$, $N$ clusters are returned, one for each data point.

### 4.2.2 Cluster Separation Measure

The segmentation of the sequence is repeated for different cutoff values $\theta_p$ of the threshold parameter $\Theta$, where $\theta_p \in \mathbb{N}$. The values $\theta_p$ for the cutoff parameter are defined as a range $a \le \theta_p \le b$. For each value $\theta_p$ a clustering result $C_p^R$ is returned. For each clustering result $C_p^R$ the value of $CSM_p$ is computed separately.

The $CSM$ is a measure of the intra-cluster compactness and inter-cluster separation of a clustering result $C^R$ [Dav79]. Compact and well separated clusters return low values of $CSM$. The $CSM$ helps to identify which values $\theta_p$ return from the data sequence clustering results with the most compact and well separated clusters.

The *CSM* derives from two primary measures:

1) The *intra cluster dispersion measure* $D_m$, which measures the compactness of clusters. The dispersion measure $D_m$ is calculated for every cluster $C_m$, $m = 1, 2, ...k$ in the clustering result $C_p^R$.
2) An *inter cluster distance* measure $M_{ij}$ or inter cluster separation measure. The inter cluster distance $M_{ij}$ is computed for any two pairs of clusters $C_i$ and $C_j$ in the clustering result $C_p^R$.

Given a clustering result $C_p^R = \{C_1, C_2, ..., C_k\}$:

- **The intra cluster dispersion** measure $D_m$ is the mean distance of the data points-members of a cluster $C_m$ around the cluster mean $Z_m$,

$$D_m = \frac{1}{N_m} \sum_{j=1}^{N_m} | t_j - Z_m | \qquad \textbf{(4-1)}$$

, $N_m = | C_m |$ is the size of the cluster (number of data points in a cluster) and

$$Z_m = \frac{1}{N_m} \sum_{j=1}^{N_m} t_j \qquad \textbf{(4-2)}$$

is the cluster mean for cluster $C_m$, where $m = 1, 2, ..., k$ . $Z_m$ is also the point representation of each cluster $C_m$ that is used for computing the distance between two clusters. The dispersion measure for single-member clusters is equal to zero, as $Z_m$ and $t_j$ are identical (Figure 4-2).

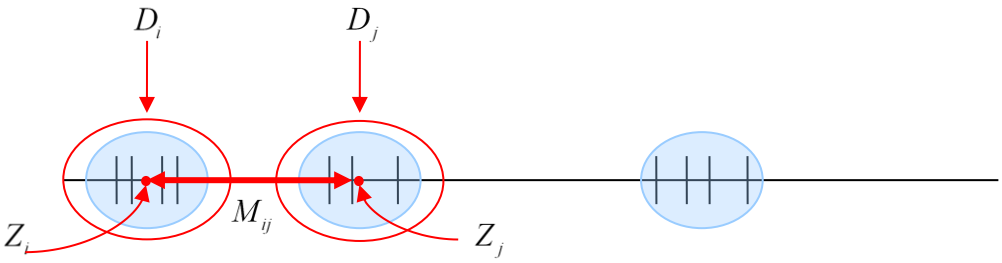



**Figure 4-2: Cluster Separation Measure**

- **The inter cluster distance measure** $M_{ij}$ is defined as the Euclidean distance between the point representatives $Z_i$ and $Z_j$ of any pair of clusters $C_i$ and $C_j$ in $C_p^R$ (Figure 4-2).

$$M_{ij} = |Z_i - Z_j|,\ i = 1,2,...k \text{ and } j = 1,2,...k \qquad \textbf{(4-3)}$$

For a clustering result $C_p^R$ that contains $k$ clusters, a $k \times k$ symmetric matrix $M$ is produced that contains the pairwise inter cluster distances $M_{ij}$ for a given clustering result (all elements on the diagonal of $M$ are all equal to zero).

Given the above, the intermediate measure $R_{ij}$ for any pair of clusters $C_i$ and $C_j$ in $C_p^R$ is given by:

$$R_{ij} = \frac{D_i + D_j}{M_{ij}} \qquad \textbf{(4-4)}$$

For each cluster $C_i$ the separation measure equals to the highest value among the separation measurements between the cluster $C_i$ and any other cluster $C_j$ in the clustering result $C_p^R$.

$$R_i^{'} = \max\{R_{ij},\ j = 1,2,..k,\ j \neq i\},\ i = 1,2,...k \qquad \textbf{(4-5)}$$

The above measures are calculated only for the upper triangular of the M matrix.

For a clustering result $C_p^R$, the **cluster separation measure CSM** is given by the average of the intermediate measures $R_i'$:

$$CSM_p = \frac{1}{k}\sum_{i=1}^{k} R_i^{'} \qquad \textbf{(4-6)}$$

It is clear from function $R_{ij} = \frac{D_i + D_j}{M_{ij}}$ (4-4 that the more compact the clusters are the smaller the values $D_i$ and $D_j$ will be, which will reduce $R_{ij}$. Also, the more separated the clusters $C_i$ and $C_j$ are, the larger the distance $M_{ij}$ will be, which again will reduce $R_{ij}$. Therefore, low values of *CSM* indicate compact clusters that are

well separated from each other. This in turn suggests that the corresponding cutoff value $\theta_p$ returns desirable segmentations.

#### *4.2.2.1* Benefits of using *CSM* over other alternatives

The cutoff parameter $\Theta$ that returns the lowest CSM among all other values in the range is selected. For making a choice it is important to have clear indication on which value of the cutoff parameter performs best.

An alternative method to the CSM is to plot the number of clusters formed for each value of the cutoff parameter in the range of $\Theta$ [Tsa83]. The plot of number of clusters indicates the best value for $\Theta$ by the "knee" (see Figure 4-3a). In this graph the vertical drop of cluster count versus $\Theta$ changes into an almost horizontal curve. This is the "knee" of the plot and an indication that this location is a good value for $\Theta$. The knee in the plot suggests that the data points have been allocated to the "right" amount of clusters as the clustering result remains unchanged for a long range of values of $\Theta$. However, the criterion of *cluster count* does not perform always that well. In a different example seen in Figure 4-3b, the curve does not provide any clear indication as to where a good value for the cutoff criterion is. In the second example the cluster count continues to decrease as the values for $\Theta$ increases, without any "knee" becoming visible in plot.

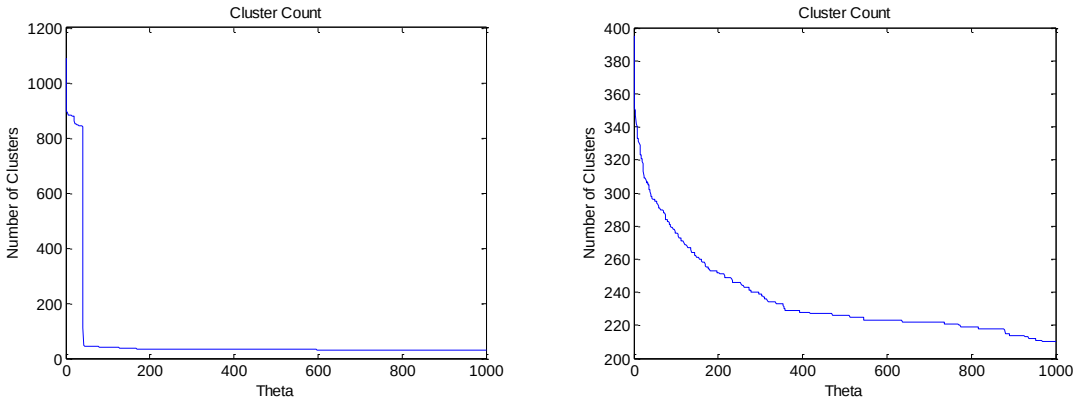



**Figure 4-3 Cluster count plots (a left, b right)**

Obviously the cutting point in the second figure is subject to the analyst's decision. Compared to the cluster count the CSM provides clear indications. In Figure 4-4 the *CSM* is plotted for the same sequence and over the same range of values for $\Theta$ as for the cluster count criterion in Figure 4-3b. The lowest values for *CSM* can be clearly identified. A CSM value is computed for every clustering result given a value of $\theta$. For $\theta = 0$ the CSM is equal to zero as for this cutoff value each cluster contains only one data point and therefore the intra dispersion measure is null. When $\theta > 0$ the CSM clearly indicates that the best result is that for $\theta = 4$ (area of red rectangle in Figure 4-4). A magnification of the red rectangle is shown in Figure 4-5.

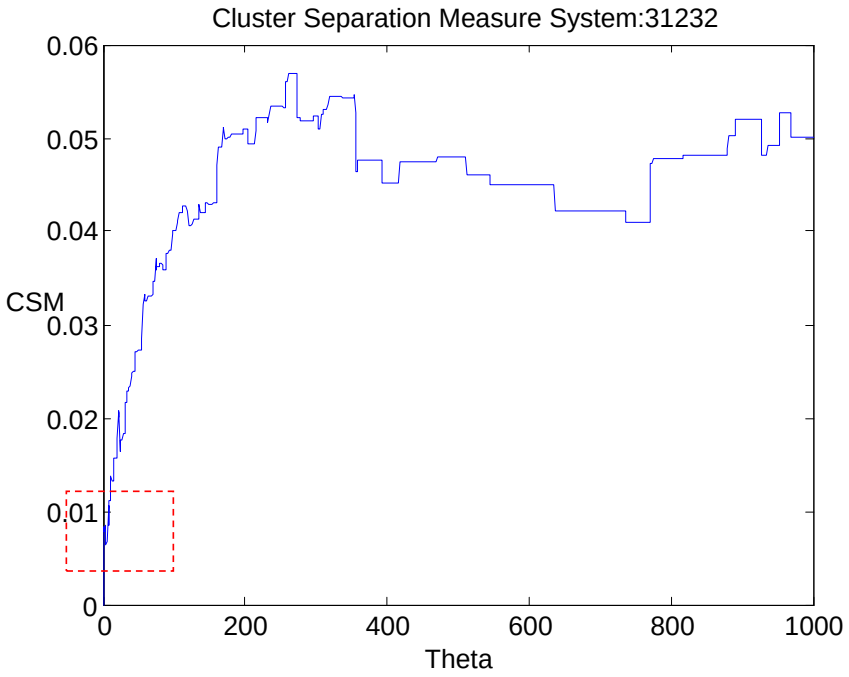



**Figure 4-4 Cluster separation measure vs. Theta**

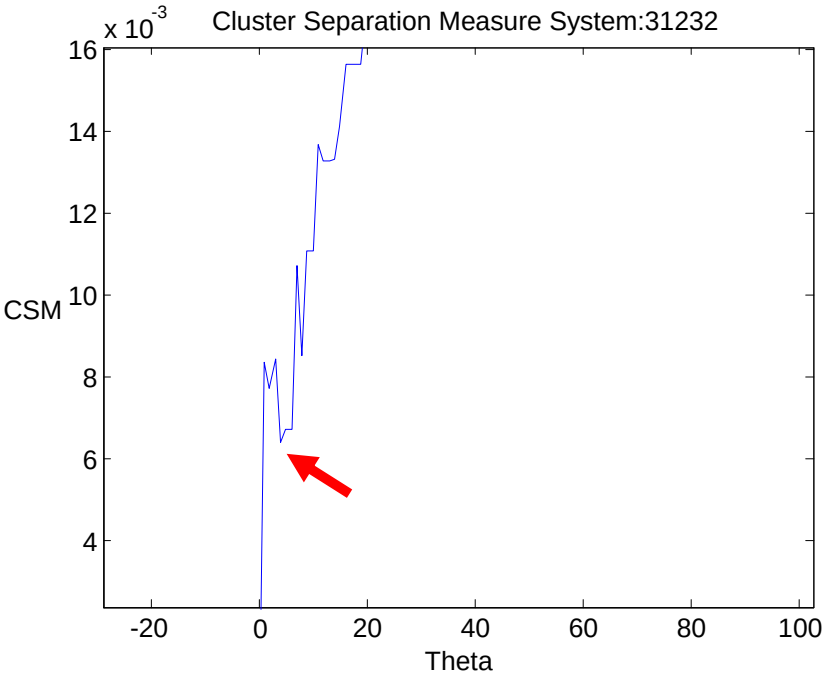



**Figure 4-5 Minimum CSM value magnified from Figure 6**

The *CSM* is not an absolute measure of good segmentation. The CSM indicates the best clustering results among the set of clustering results $C_p^R$ that have been obtained by segmenting the data sequence $S$ within a given range of $\Theta$. Using the CSM values to compare the segmentation results for sequences is not informative.

### 4.2.3 Robustification the cutoff parameter

To reduce the risk of truncation when the algorithm is applied in the field, the value of the cutoff parameter is adjusted to tolerate variation in the structure of subsequences. In the fault injection experiment it was shown that distances between successive traces can vary although the overall length of the subsequence can remain

approximately the same. For different instances of the same type of subsequence (type defined by the injected fault) the average distance between the traces can remain approximately the same; however the configuration of the points within the subsequence's length can vary. The configuration of traces is specific for every instance of the subsequence, and since the sample sequence consists of such instances the cutoff value as it is set with the help of the CSM value, is subject to the particular trace configurations of the sampled sequence. The risk of truncation is caused if the sampled sequence leads to a "tight" fit of the cutoff parameter. To reduce this risk more instances of the sampled sequence can be simulated where the configuration of the traces is different. The simulated instances of the original sequence are called variants. In variants the subsequences are manifested in such a way that the average distance between traces remains the same but the different configurations of traces in the subsequence allow the distance between successive traces to take extreme values. The cutoff value is then assessed how well it can perform on the variants.

The variation in the sequence is simulated using the resampling method (RM) of Levine [Lev01]. The original idea behind Levine's method [Lev01] is to assist the clustering operation in the presence of noise (random data entries). Here it is used to produce variants $S'$ of the original sample data sequence $S$. The variants $S'$ are then segmented using the same clustering algorithm and the clustering results are used to assess how robust $\theta_p$ is to variation in the structure of subsequences. The resampling method is described in 4.2.3.1. The parameterization of the resampling method is described in 4.2.3.2. The hCSM and the robustification of the cutoff value are described in 4.2.3.3.

#### 4.2.3.1 Resampling method

Variants $S'$ are produced out of the original sequence $S$ by randomly selecting data points out of $S$ with a *thinning probability* $f$. For every data point in the original data sequence $S$, there is a probability $f$ that it will be selected during resampling. The variant $S'$ is therefore a "thinned" or "diluted" version of the original data sequence $S$.

In Figure 4-6 an illustration of the resampling method is shown. The original sequence $S$ contains three subsequences. Each subsequence consists of a number of traces (vertical lines). The red lines indicate the traces that are selected in a sampling instance with a thinning probability of $f = 0,5$. With this thinning probability, half of the original data points will be selected on average to form the variant $S'$. The higher the thinning probability $f$ is, the more data points of the original sequence $S$ will be present in the variant $S'$.

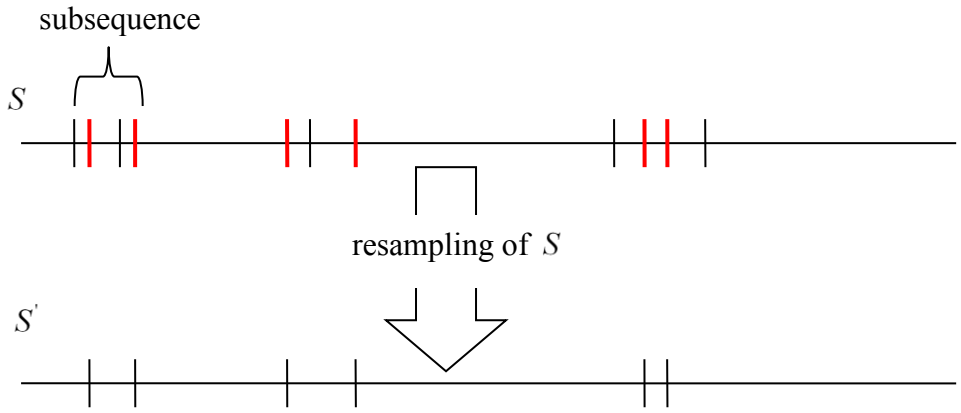



**Figure 4-6 Resampling data sequence *S***

Once a variant has been produced both $S$ and $S'$ are segmented. First the original data sequence is segmented for the cutoff value $\theta_p$, producing a clustering result $C_p^R$. Then the resampled data sequence $S'$ is processed using as cutoff value $\theta_p' = \frac{\theta_p}{f}$ and a clustering result $C_p^{R'}$ for the variant is produced. The two clustering results are the compared to find out which data points that **are members of the same cluster** in the clustering result $C_p^R$ of the original data sequence $S$, **also remain members of the same cluster** in the clustering result $C_p^{R'}$ of the resampled data sequence $S'$. The agreement between the membership of data points for the results of the original sequence and its variant is reflected by the *measure of merit* $M_L$. If all data points that are members of a cluster in $S$ are also members of the same cluster in $S'$, there is a total agreement between the two clustering results. This agreement results to a measure of merit $M_L = 1$. If the measure of merit is lower than 1, it suggests that there was not absolute agreement between the membership of data points in clusters in the original clustering result and the result of the variant sequence. This in turn suggests that the value for the cutoff criterion is not less to variation. The lower the *measure of merit* $M_L$ is, the lower the robustness of cutoff value $\theta_p$.

The process is repeated $k$ times, by producing $k$ variants of $S$ using the same thinning probability. Each variant is segmented and the clustering result is compared again the original clustering result to compute the measure of merit. Averaging $M_L$ over all $k$ gives a representative measure of agreement for the clustering results produced by $\theta_p$.

The resampling method is applied on the data sequence over the selected range of values for $\theta_p$. An example of how $M_L$ varies over the range of $\theta_p$ is shown in Figure 4-7. The higher the value of $M_L$ is (max is 1), the more robust $\theta_p$ is to the variation in the structure of subsequences.

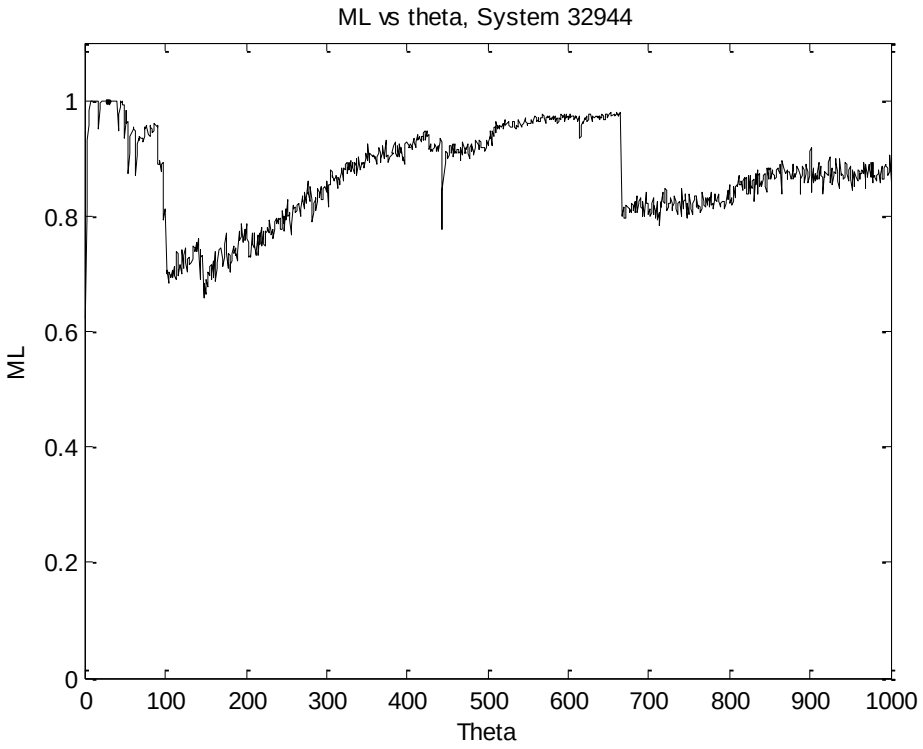



**Figure 4-7 Robustness measure $M_L$ vs. Theta**

The above described resampling method allows the creation of random variants of subsequences every time the original sequence is sampled. The figure of merit then indicates how robust $\theta_p$ is to the induced level of variation. To select the value $\theta_p$ that returns a good segmentation result and is also robust to variation the two measures CSM and $M_L$ are combined to produce the hybrid CSM (hCSM).

#### 4.2.3.2 Choosing the thinning probability

At this point it is required to decide which value of $f$ serves best the robustification of the cutoff parameter. Since the main target of the robustification is to reduce the risk of truncation, it is of interest to produce at least one variant that contains a distance between successive traces (DBST) that exceeds the maximum distance between successive traces that is found in the original subsequence. Variants that contain at least one such DBST will lead for a given cutoff value to the truncation during the clustering of variant sequence, which in turn will reduce the figure of merit $M_L$ for that cutoff value.

The variants of the original subsequences are produced in such a way that they retain their relative structure. This structure is defined as the subsequence's density i.e. the mean distance between successive traces (mDBST). This restriction assures that the variants are effectively variations of the original subsequence where the traces are rearranged within the same interval length as the original subsequence.

Given the above the aim is in choosing a value of the thinning probability, for which the variants:

a) Retain the density of the original subsequence.

b) Produce DBST that exceed the DBST in the original sequence. This criterion represents the effectiveness of the thinning probability in the robustification.
c) Have a high chance of producing DBST that exceed the DBST in the original sequence. This criterion represents the efficiency of the thinning probability in producing DBSTs that will exceed the maximum DBST of the original subsequence.

To find out which value of the thinning probability $f$ can achieve best the above criteria, an empirical analysis is conducted. Original subsequences are produced for different density levels mDBSTo = [3,5,8,10,16,20,40,50,65,95,180,345]. For each level of $mDBST_O$ variants of the original subsequence are produced using the resampling method. The resampling uses a range of values of the thinning probability starting from $f_1$=0.05 and increasing by a step of 0.05 up to $f_{19}$=0,95. For every new variant that is produced by resampling the distances between points in the subsequence are rescaled with the thinning probability$f$. This resembles the effect of adjusting the cutoff value as in the resampling method.

For each level of $mDBST_O$ 100 subsequences are produced. For each level of the thinning probability the original subsequence is resampled 5000 times. For each level of $mDBST_O$ the following measures are averaged over the 100 samples:
a) The mean distance between traces in the original subsequence $mDBST_O$
b) The largest distance between successive traces in the original subsequence $maxDBST_O$

For each level of the thinning probability $f$ the following measures are averaged over the 5000 samples:
a) The mean distance between traces $mDBST_V$of the variants
b) The largest distance between successive traces, $maxDBST_V$ in the variants
c) The number of DBST in the variants that are greater than $maxDBST_O$.

To examine how the thinning probability performs for each one of the stated criteria, the following measures are computed for each level of $mDBST_O$and for each value of the thinning probability$f$:
a) The ratio $\frac{average\ mDBST_O}{average\ mDBST_V}$
b) The ratio $\frac{average\ maxDBST_V}{average\ maxDBST_O}$
c) The ratio $\frac{average\ Number\ of\ maxDBST_{Vi}>maxDBST_O}{average\ Total\ number\ of\ DBST_V}$

#### *4.2.3.2.1 Retain the density of the original subsequence*

Since the $mDBST_O$ of the original subsequence has multiple levels, the criterion is tested using the ratio $\frac{average\ mDBST_O}{average\ mDBST_V}$. The criterion is met when the ratio is 1, which suggests that the mDBSTv of the variants is approximately equal to the mDBSTo of the original subsequence. In Figure 4-8 the ratio is graphed using a contour plot. The vertical axis represents the levels of average $mDBST_O$ and the horizontal axis the values of the thinning probability. The coloring of the lines in the plot indicates how close the ratio is to 1. The color coding is explained in the color scale on the right side of the figure. Dark brown and red lines indicate the areas where the ratio is equal to or very close to 1.

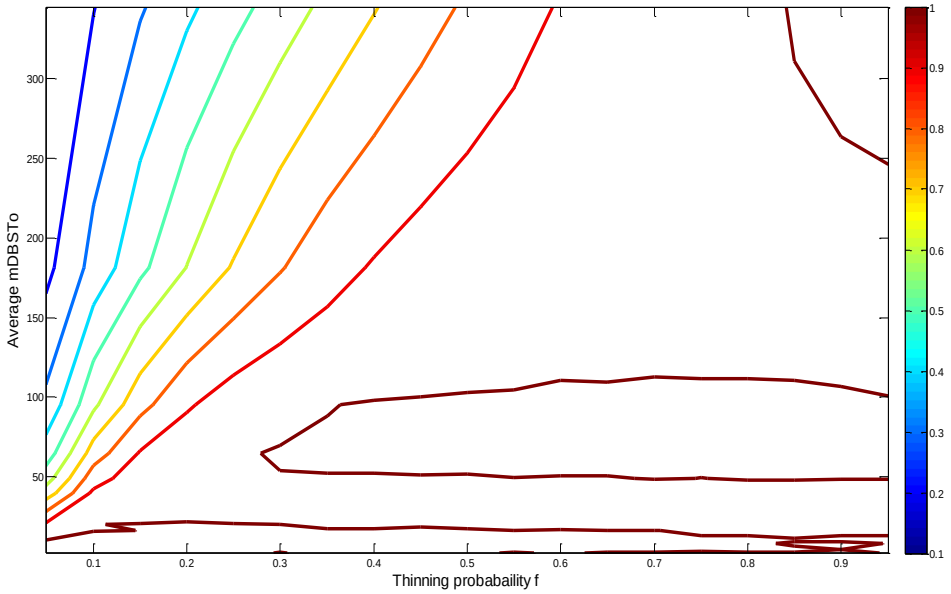



**Figure 4-8 Ratio of average mDBSTo over average mDBSTv for all f**

It can be seen that for dense subsequences (levels of $mDBST_O$ [3,5,8,10,16,20] ) the thinning probability performs well (the ratio is close to 1). As the $mDBST_O$ increases >20 the performance of the low values of the thinning probability begin to drop, but the performance remains on good levels for higher values of the thinning probability. The area where the performance of the criterion is satisfactory is on the right side of the red contour line that starts at the bottom left corner and ends at the right of the middle of the top of the figure. Within this area the thinning probability performs well and is meeting the first criterion for all levels of mDBSTo.

#### *4.2.3.2.2 Produce DBST that exceed the DBST in the original sequence*

Figure 4-9 shows the three dimensional plot of the ratio $\frac{average\ maxDBST_V}{average\ maxDBST_O}$ over all levels of $mDBST_O$and all values of the thinning probabability $f$. For every level of the average $mDBST_O$, the performance of the ratio is assessed over the range of values of the thinning probabilities. Large values of the ratio are preferred as they indicate large DBST in the variants that exceed $maxDBST_O$ . For a given level of average mDBSTo, the location where the ratio is the highest, is the best choice of the thinnning probability. The blue curve in Figure 4-9 indicates for each level of average $mDBST_O$the location in the range of thinning probabilities where the ratio takes its highest value. The red curve in Figure 4-9 is the projection of the blue curve on the $f$-mDBST plane. The red curve makes the reading of the best locations of the thinning proabbaility easier.

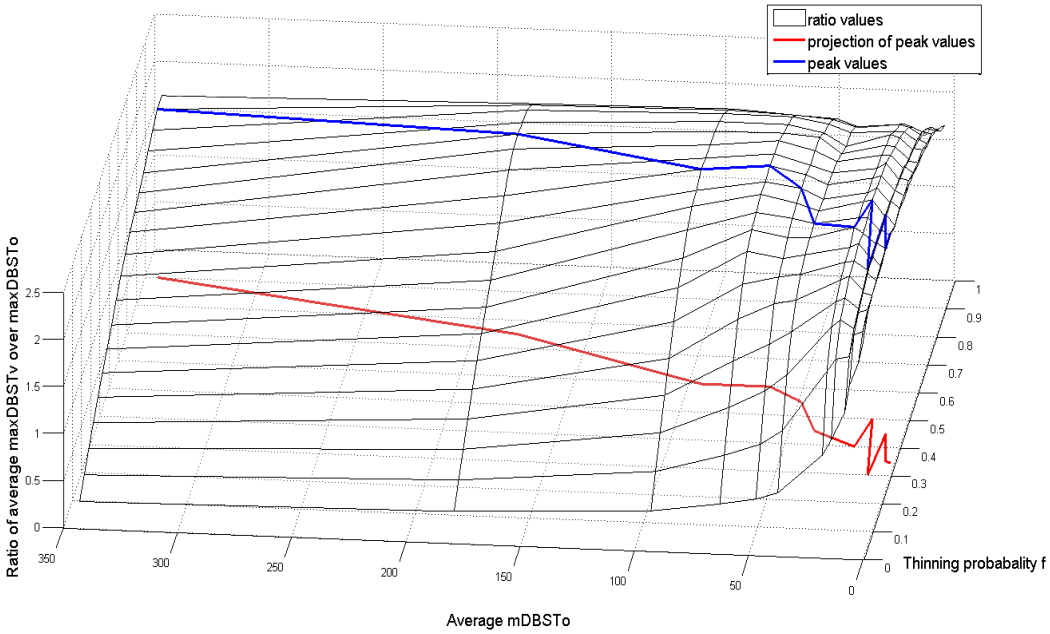



**Figure 4-9 Ratio of average maxDBSTv over maxDBSTo for all f**

The red curve shows that the most effective (maximum) values of the thinning probability for dense subsequences are between 0.3 and 0.5. As the density of the original subsequence reduces, mDBSTo>50, the most effective values of the thinning probability are found in the high end of the range.

The red curve of Figure 4-9 is shown Figure X on the $f$-mDBST plane. For dense subsequences the average $maxDBST_V$ can be larger than average $maxDBST_O$by a factor of two. The ratio drops just below the factor of 2 for lower density levels but still performs keeping in mind that this is an average value and a single such large value is enough to cause the truncation of the variant subsequence.

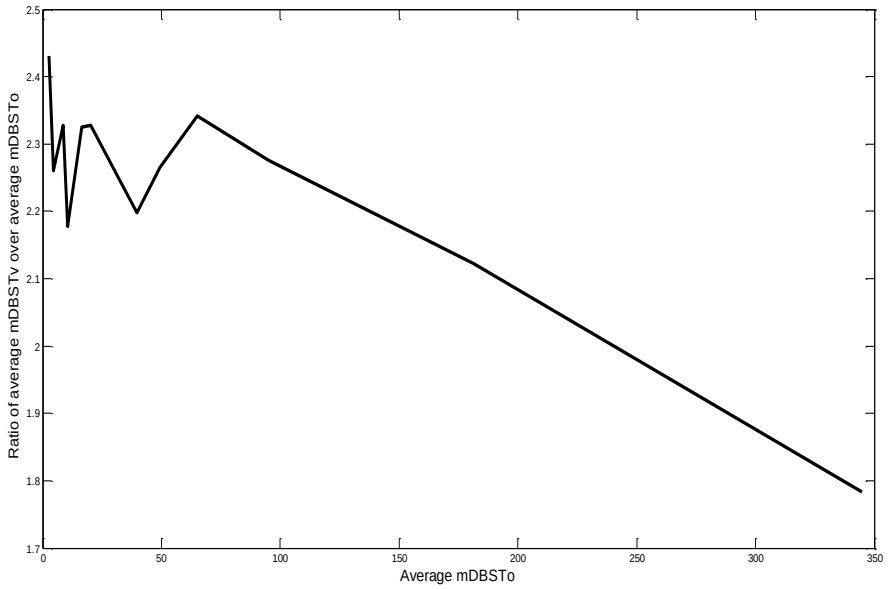



**Figure 4-10 Best values of ratio over average mDBSTo**

#### *4.2.3.2.3 High chance of producing DBSTs that exceed the maxDBST in the original sequence*

The third aspect of this experiment is the efficiency with which the thinning probability in producing such variants that contain DBSTs that exceed the $maxDBST_O$ of the original subsequence. In Figure 4-11the results of the analysis are shown. The figure shows the ratio of the average number of DBSTv that are greater than the $maxDBST_O$ over the total number of DBSTv in the variant. The blue curve indicates the peaks of the ratio i.e. the locations where the ratio is the highest. The red curve is the projection of the blue curve on the $f$- mDBST plane.

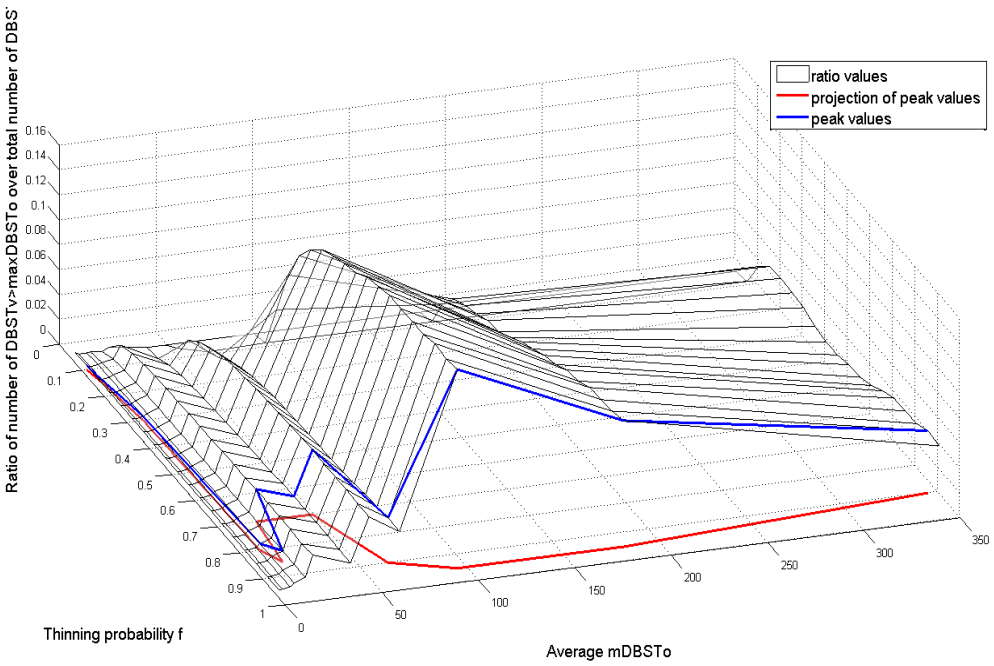



**Figure 4-11 Ratio of number of DBSTv that exceed maxDBSTo over total number of DBSTv in variants**

The efficiency of the resampling method is highest for values of the thinning probability other than the values that score best for effectiveness. However the ability to produce a $maxDBST_V$ that exceeds the $maxDBST_O$ is more important than the efficiency with which these large distances are being produced. Moreover when if the thinning probability is chosen based on effectiveness, the efficiency of those values is only marginally lower than if the thinning probability is chosen based on best efficiency. In Figure 4-12 the black curve indicates the values of the ratio where it peaks in the ranges of the thinning probability and the red curve indicates the values of the ratio where the thinning probability is most effective.

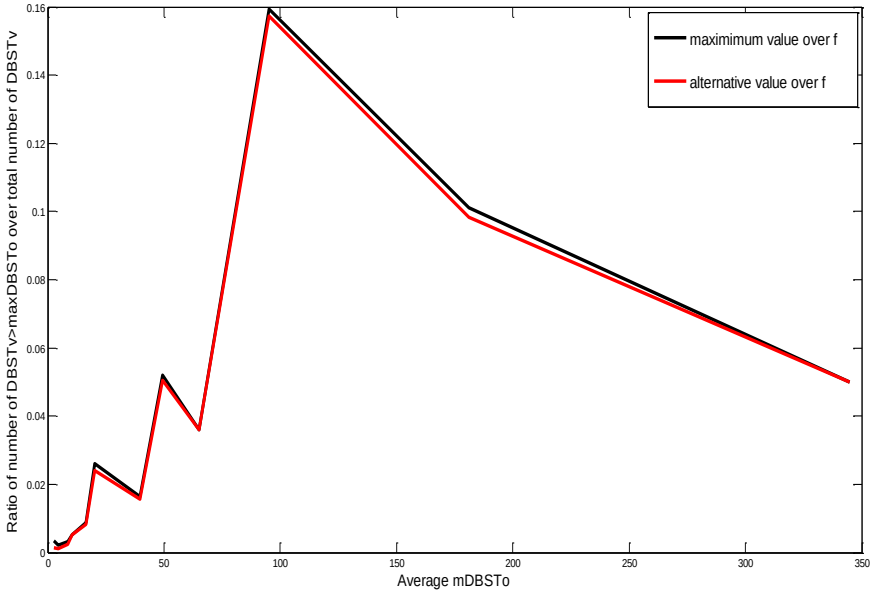



**Figure 4-12 Efficiency of the thinning probability in producing large DBSTv**

#### *4.2.3.2.4 Proposed thinning probabilities*

Based on these results the value for the thinning probability is selected in accordance to the density of the subsequences, as they are produced for a given value of the cutoff criterion. The table below shows these values:

| mDBST | Value of thinning probability |
|---|---|
| 3 | 0.3 |
| 5 | 0.3 |
| 8 | 0.4 |
| 10 | 0.4 |
| 16 | 0.4 |
| 20 | 0.4 |
| 40 | 0.4 |
| 50 | 0.5 |
| 65 | 0.6 |
| 95 | 0.6 |
| 180 | 0.7 |
| 345 | 0.9 |

**Figure 4-13 Proposed values for the thinning probability for different levels of mDBSTo**

### 4.2.3.3 Hybrid-CSM, a criterion for choosing robust $\theta_p$

To combine the information obtained from the resampling method together with the *CSM*, a new measure is defined, the *hybrid cluster separation measure*:

$$hCSM = \frac{CSM}{M_L} \qquad \textbf{(4-7)}$$

The hCSM is applied on the range of values of $\theta_p$ that have been indicated by the CSM to return the most compact clusters. This measure has the property of penalizing CSM for the areas of $\theta_p$ that do not score well in robustness. Areas that score perfectly well in robustness ($M_L = 1$) retain their original CSM value. For the *hCSM* the selection of the best clustering result is done in the same way as for the CSM i.e. the lowest value indicates the best clustering result.

To illustrate how $M_L$ transforms CSM to *hCSM,* a data sequence is processed and both measures are obtained. In Figure 4-7 the entire range of $\theta_p$ is shown. The black line represents the CSM and the red line the hCSM. Areas of $\theta_p$ that did not perform well during the resampling method, are penalized by the low value of $M_L$

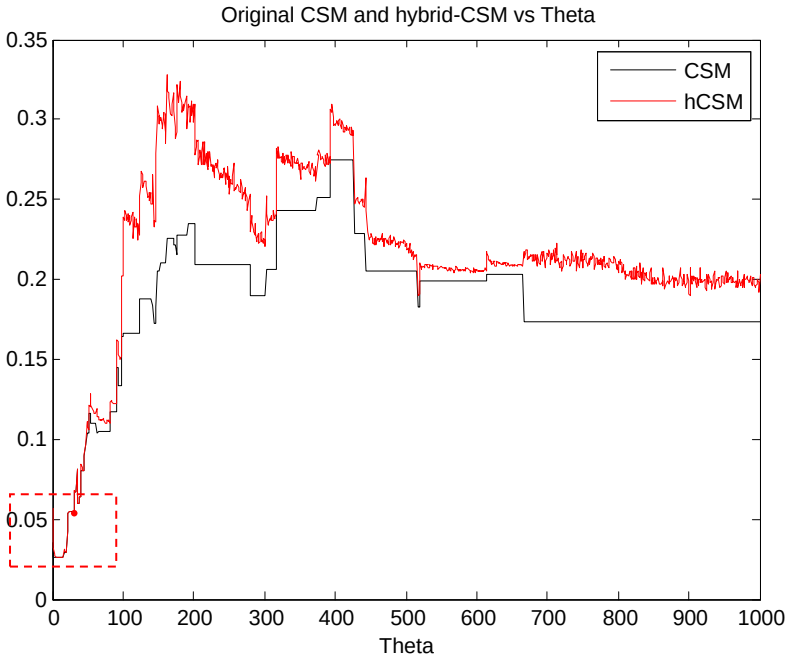



**Figure 4-14 Original CSM and hybrid-CSM vs. Theta**

The best segmentation results are found at the low end of the range (red rectangle). In Figure 4-15 a magnification of the area in the red rectangle of Figure 4-7 is shown. The original *CSM* (black line) suggests that the best values for $\theta_p$ is the narrow range between the values $[2,16]$. The *hCSM* however suggests that the best range is $[8,16]$. The *hCSM* suggests obviously a region with higher cutoff values, because larger value of the cutoff criterion can perform better after the thinning of the original sequence. The cutoff values in the range $[2,7]$ did not score well in robustness. The value $M_L$ was low, leading to an increase of *hCSM*.

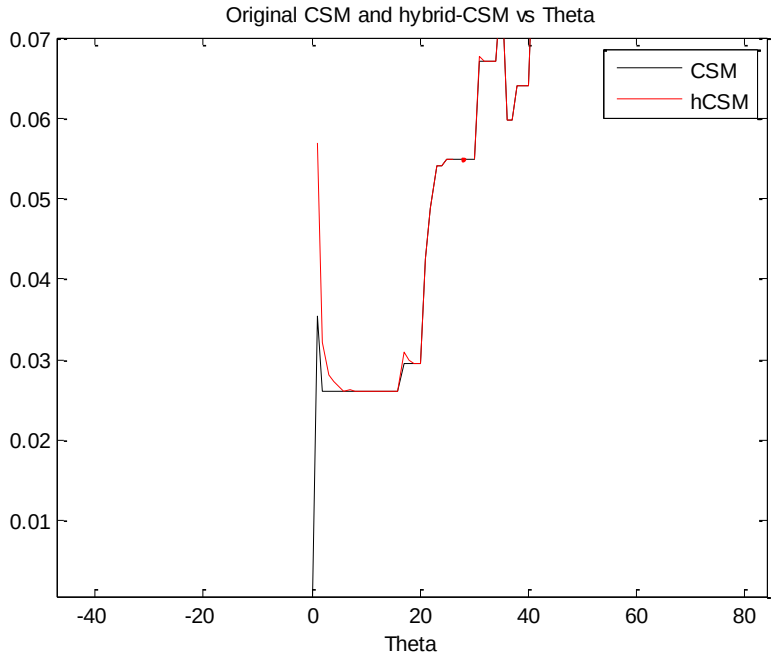



**Figure 4-15 Original and hybrid-CSM magnified at area of interest**

With the above robustification of the cutoff value the risk of truncation is reduced.

### 4.2.4 Collision probability (CP)

The problem of collision was discussed by Hansen [Han92] who described an empirical approach in estimating the probability of collision (CP) given a clustering result. The collision probability is most relevant of the application of the algorithm in the field. Taking CP into account when defining the cutoff parameter will reduce the probability of merging together subsequences that are created by random failure events that occur closely after each other.

Given that *X* is a random variable for the inter arrival time of the error process and *L* a random variable for the length of the subsequence resulting from an error, a *collision* will take place if the interval *Z=X-L,* that is the time between the end of one subsequence and the beginning of the next subsequence, is smaller than the cutoff value $\theta_p$ .The probability of *collision* can then be estimated by [Han92]:

$$\Pr(X - L < \theta_p) = 1 - e^{-\lambda_F \vartheta_p} \left( \sum_{i=1}^{n} p_i^* e^{-\lambda_F l_i} \right) \qquad \textbf{( 4-8)}$$

Where $p_i^*$ the probability for length $l_i$ occurring. The values for $p_i^*$ are empirically estimated from the collection of clusters in the clustering result $C_p^R = \{C_1, C_2, ..., C_k\}$. Every cluster $C_i$ in $C_p^R$ has a cluster length $l_i$, defined by the two most distant members within that cluster $l_i = t_{\max} - t_{\min}$ ,where $l_i \geq 0$ ($l_i = 0 \, if \, t_{\max} \equiv t_{\min}$. i.e. a singleton cluster).

The failure intensity $\lambda_F$ of the physical process is assumed to be exponentially distributed. Because it cannot be measured directly it is estimated based on the clustering result. The method is described in detail in [Han92]:

**The smaller the value $\theta_p$ of the cutoff criterion, the lower the CP is.**

To illustrate how CP affects the choice of $\theta_p$, an example is provided. For simplicity reasons in the example CSM is used instead of *hCSM*. The CP applies on both measures in the same manner.

The objective is to reduce *CP* given a range of $\theta_p$ that was indicated by CSM as returning the best segmentation of the sequence. CP tends to favor the left end of the proposed range. This is because the smaller the value of the cutoff criterion is, the lower CP is. In Figure 4-16 the *CSM* values from the segmentation of sequence is shown. The best region for $\theta_p$ is lies between the two vertical lines as shown in the graph. The *CP* criterion suggests the lowest value of the range, where the *CP* is the lowest.

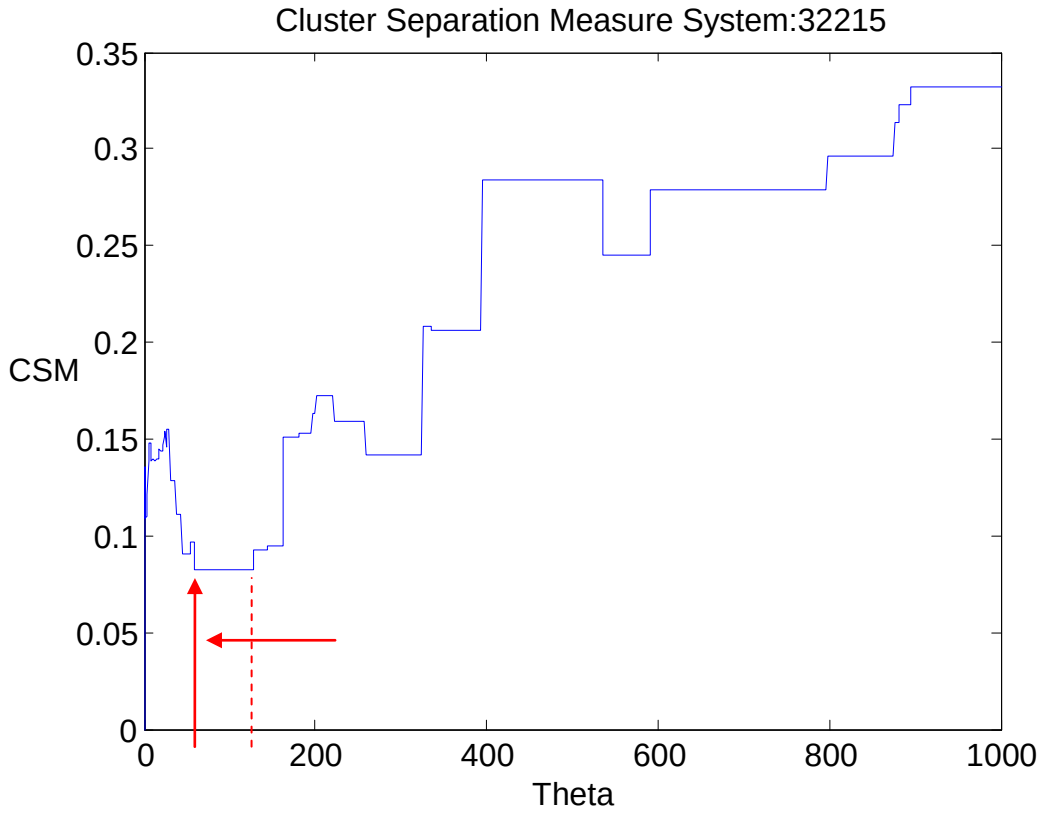



**Figure 4-16 Effect of CP on the selection of the value of the cutoff criterion**

### 4.2.5 Internal Criteria Validation

The sections so far discussed the process of selecting an appropriate cutoff value using *hCSM* as the primary and the *CP* as the secondary criterion. The result of that process is a cutoff value $\theta_u$ for which the clustering result satisfies best three requirements:

1. Best possible segmentation result of the sample sequence
2. Robustness to variation of subsequences (reduced risk of truncation)
3. Low collision probability

This section discusses the last step of the segmentation process, which is the validation of the cluster result $C_u^R$.

#### 4.2.5.1 Random position hypothesis testing

The aim of the hypothesis test is to examine whether the clusters obtained from the segmentation of the sequence *S* are due to the non-random arrangement of traces in the sequence. The test examines the likelihood of obtaining similar clusters from a sequence with the same number of traces, but where the traces a randomly arranged over the length of the sequence. If such likelihood is small, then the clustering result obtained by the segmentation, is a *sensible* result, and combined with the use of CSM it is the *best sensible* result.

The random positioning hypothesis assumes that "*All arrangements on K vectors in a specific region of the l-dimensional space are equally likely to occur"*[The06][Jai88]. In the case of traces this assumption can reformulated as: "*All N traces found in the sequence S, are positioned randomly*". If the sequence has a random structure, i.e. the traces are arranged randomly over the length of *S* then the clustering result that is obtained by the segmentation of *S* is coincidental. If so, a similar result can be obtained by other randomly structured sequences. The hypothesis test examines how likely it is to obtain from randomly generated sequences similar results, as the results obtained from sequence *S*. The null hypothesis is formulated as following:

*$H_0$: The clustering result $C_u^R$ that is obtained by the segmentation of $S$, which contains N data points ,and by using the cutoff value $\theta_u$, is equally likely to be obtained from a data sequence $S_R$ of the same length as $S$*, where *N data points are positioned randomly.*

the alternative hypothesis states:

*$H_1$: The clustering result $C_u^R$ that is obtained by the segmentation of $S$, which contains N data points, and by using the cutoff value $\theta_u$, is unlikely to be obtained from a data sequence $S_R$ of the same length as $S$*, that *contains N data points that are positioned randomly.*

The test consists of the steps:

1. Generate a sequence $S_R$ of the same length as the sequence *S*, and which contains *N* points uniformly distributed over its length.
2. Use an appropriate test statistic to compare the clustering result obtained by segmenting S against the clustering result obtained by segmenting $S_R$
3. Perform the hypothesis test using the empirical distribution of the test statistic.

#### 4.2.5.2 Test statistic

To test the hypothesis the normalized $\hat{\Gamma}$ statistic is used. The normalized $\hat{\Gamma}$ statistic is a normalized version of Hubert's $\Gamma$ statistic [The06]. The $\hat{\Gamma}$ statistic measures the

correlation between two matrices. Large absolute values of $\hat{\Gamma}$ are an indication of good agreement between the elements of the matrices.

For a given cutoff value, the statistic is computed for the sequence and clustering result and is compared against the distribution of the statistic of the random position sequences and their clustering results.

The sequence is represented by a proximity matrix. The proximity matrix contains the information on the distances of a data point to all other data points in the sequence. The proximity matrix is a symmetric matrix where the values of the diagonal are equal to zero (distance of a data point to itself). The clustering result $C_u^R$ is represented by the *dis-connectivity matrix* $\overline{T}_C$. The *dis-connectivity matrix* $\overline{T}_C$ is a cluster membership representation of the data points in the sequence. The pairs of data points that are *not* members of the same cluster are indicated by the value of 1.

The normalized $\hat{\Gamma}$ statistic measures the agreement between the proximity matrix and the dis-connectivity matrix $\overline{T}_C$ . Large values of $\hat{\Gamma}$ suggest that the clustering result $C_u^R$ agrees with the inherent structure (relative distances of data points) of the sequence.

The matrices $P$ and $\overline{T}_C$ as well the test statistic $\hat{\Gamma}$ are described in more detail in Appendix C.

#### 4.2.5.3 Null hypothesis testing

The hypothesis testing is based on the comparison of the $\hat{\Gamma}_u$ statistic obtained from the sequence against the empirical distribution of the statistic $\hat{\Gamma}_r$ obtained by the random positioning sequences.

Random positioning sequences are produced *k* times, by distributing uniformly N data points over an interval of length equal to the sampled sequence *S*. For every random positioning sequence $S_{R_i}, i = 1, 2, ..., k$, the corresponding proximity matrix $P_{Ri}$ is produced. Also, the sequential algorithm is applied on every $S_{R_i}$ with the same value $\theta_u$ of the cutoff criterion. Given the clustering result the dis-connectivity matrix $\overline{T}_{CRi}$ is produced for each sequence and the statistic $\hat{\Gamma}_r$ is computed. The set of $\hat{\Gamma}_r = \{\hat{\Gamma}_1, \hat{\Gamma}_2, ..., \hat{\Gamma}_k\}$ provides the empirical distribution of the test static under the null hypothesis of random positioning $P(\hat{\Gamma}_r \mid H_o)$

At a significance level $\alpha$, the *null hypothesis* is rejected (accepted) if $\hat{\Gamma}_u$ is greater (smaller) than $(1-\alpha)k$ of the $\hat{\Gamma}_r$ values.

## 4.3 Discussion and Conclusions

In this chapter the first step of the transformation process was described. The segmentation step operates on the sequence and defines the subsequences. The subsequences are representations of physical events. The parameterization of the segmentation algorithm is done using a sampled sequence from a single system. The parameterization is based on the definition of the most compact subsequences possible in the sampled sequence. This is based solely on the temporal information in the sequence.

The hCSM that is proposed in this chapter allows the selection of values for the cutoff parameter that perform can reduce the risk of truncation due to variation in the subsequences, when the algorithm is applied in the field. Additionally the value of the cutoff parameter is adjusted to reduce the risk of collision using the collision probability criterion.

The clustering result is validated whether it is likely to be obtained from a sequence with no particular data structure. The segmentation method used in this research falls under the category of unsupervised clustering. The random positioning hypothesis aims to validate that the detected subsequences are sensible clusters based on the arrangement of traces in the sequence.

The proposed method meets the requirement of detecting subsequences by using only the information found in the sequence. It also prepares the method for real life applications with the robustification. The segmentation method described in this chapter is demonstrated in chapter 7.

# Chapter 5

# 5 Tagging of subsequences and tag matching

In this chapter the second step of the transformation phase is described. This step involves *Tagging* and *Tag matching.* The reduction of the size of data, without the loss of any relevant information, is one of the main goals of this research. This objective will be addressed with this step.

With the segmentation of the sequence, the subsequences are identified and consequently so are the points in time where physical error and recovery events are believed to have occurred. One form of data reduction comes with the appropriate point representation of subsequences. With the point representation, a subsequence that can consist out of multiple traces each having its own temporal location, is represented by a single point in time. This way the temporal representation of the physical event is simplified.

Besides the time of occurrence of a system failure or a system recovery, there is also the semantic aspect of the physical event. The semantics of traces can provide information about *the nature (*what and where*)* of the events that have occurred. The semantics within a subsequence can indicate the location and the cause of the error by the qualitative information of the attribute fields of traces such as *description*, *logging unit, system state* etc. In this chapter the focus shifts from the temporal to the semantic information found in subsequences. The methodology presented aims in reducing the amount of representations needed to convey the same information on error and recovery events.

Firstly the semantics within the subsequences are reduced to the minimum necessary and they are ordered. A subsequence can contain multiple traces of the same type i.e. same *ic.* Replicates of a type of trace contain the exact same semantic and have therefore no additional informative value. Replicates are redundant and can be removed. The order of traces in the subsequence is the result of variation induced by the error propagation and the logging mechanism. It is therefore possible to reorder the traces in the subsequence in a preferred manner without losing relevant information. The point representation, the elimination of replicates and the ordering of semantics, are defined by one method, the *tagging* of the subsequence. The *tagging* transforms a subsequence to a *tag*.

Each tag in the sequence contains a string of semantics that represents an instance of an unknown physical event. Identical tags are assumed to represent instances of the same *type* of physical event. On the basis of known relation between the semantics in a tag and the occurrence of a physical event (fault injection experiment), the more similar the tags are, the more likely it is that they represent the same physical event. Based on this assumption, a methodology is introduced that can help simplify the representation of physical events in the sequence by grouping tags into tag types according to the similarities between tags. The comparison and grouping of tag into tag types is referred to as *tag matching*. The basis of the inter tag similarity measure is a *cost function* that roots in the engineering design principle of strong coherence within and weak coupling between modules. This engineering principle sets the rules on which traces can occur in the same subsequence and which cannot. Components of

the same module can fail together thus the traces of those components can occur together and. The more often trace occur together the stronger the similarity between these traces is and vice versa.

The chapter is organized as follows: The tagging operation is described in 5.1. The tag matching is described in 5.2. The chapter closes with discussion and conclusions in 5.3.

## *5.1 Tagging Subsequences*

For the representation of the semantics the identification code *ic* of each trace is used. An *ic* is a unique alphanumerical identifier that represents all static semantic information of a trace

The *tagging* operates on the semantic information found in the traces in the subsequence. To facilitate the description of the operation the notation for traces used in Chapter 4 is extended to include the semantic information: a trace is represented by the tuple $(A,t)$, where $A$ is the semantic information of the trace and $t$ is the time of occurrence. The semantic $A$ can take takes values from a finite set of identification codes $IC = \{ic_1, ic_2, ..., ic_q\}$.

The tagging eliminates the replicated of a semantic $ic_i$ in the subsequence and arranges semantics in lexicographical order. The tagging also transforms the subsequence to a point representation. First background information regarding the variation found in the semantics is provided in 5.1.1. Then the tagging operation is described in 5.1.2.

### 5.1.1 Variation in order and frequencies of traces in subsequences

The results of the fault injection experiment showed that the ordering and the frequencies of the traces found in subsequences resulting from the same fault can vary. In the example of Figure 5-1 three instances of error subsequences result from the same fault being injected into the system. For this example the semantic $A$ takes values from the low case Latin alphabetical characters ($IC = \{a, b, c, \ldots., z\}$).

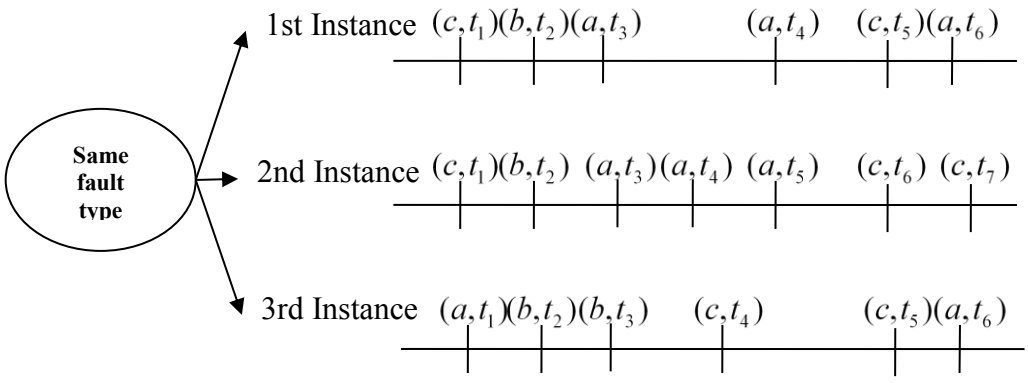



**Figure 5-1 Fault injection and result signature instances**

Although all three instances contain the semantics *a, b* and *c*, these semantics appear in different order and in different counts in each subsequence. The 1st instance contains three counts of $a$, one of $b$ and two of $c$. The second instance contains three

counts of *a,* one of *b* and three of *c*. The third instance contains 2 counts of *a*, two of *b* and two of *c*. Although the three instances of subsequences differ in the order and the count of semantics, they carry the same information, namely the occurrence of *a, b* and *c*. The replication of semantics does not bring additional informative value because the information in replicate is identical. In the fault injection experiment it was shown that the order of semantics in the subsequences can vary significantly for the same type fault. In this methodology the order is ignored and is considered to be an artifact cause by the dynamics between the components and the logging mechanism. To reduce this source of variation in the semantics of subsequences, the semantics are ordered in an appropriate manner to increase the readability of the subsequence and make the comparison of subsequences easier.

### 5.1.2 From subsequences to tags

Given a collection of subsequences $C^R = \{C_1, C_2, ..., C_M\}$ obtained from the same sequence, the tagging transforms each subsequence $C_m$, $m = \{1,2,3,..,M\}$, into a *tag* $(L_{m,} t_m)$, where $L_m$ the semantic information and $t_m$ the temporal information of the tag.

The process is taking place in two actions and is unidirectional i.e. it is not possible to retrieve the original form of the subsequence after tagging. The first step transforms the temporal information and the second the semantic information of $C_m$. With the tagging operation complete, the set of subsequences $C^R$ becomes a set of tags $\overline{C}^R$.

#### 5.1.2.1 Tagging the temporal information

The transformation maintains only one temporal location for each subsequence $C_m$. For error subsequences the *most representative* temporal location is the first trace of the subsequence because it indicates the first moment in time the physical error was detected. For recovery subsequences the most representative location is the last trace of the subsequence, as it is the closest point in time when the recovery of the system is completed. In this chapter the operation is described assuming error subsequences.

For a subsequence $C_m = \langle (A, t_i), (A, t_{i+1}), ..., (A, t_{i+n}) \rangle$, where $i = \{1,2,3,...N\}$ and $n \in \mathbb{N}$ the temporal location of $(L_{m,} t_m)$ is $t_m = \min\{t_i, t_{i+1}, ..., t_{i+n}\} = t_i$. The definition of the tag can be extended to include other temporal information of the subsequence if that is required, as for example the duration of the subsequence $d_m = t_{i+n} - t_i$, can be included to form a 3-tuple of the form $(L_{m,} t_m, d_m)$.

#### 5.1.2.2 Tagging the semantic information

For the transformation of semantic information a function *F* is defined that is transforming the subsequence $C_m$ into the tag $L_m$. The function *F* is composed out of two sub-functions $F = f_O \circ f_S$, an *order function* $f_O$ and *set function* $f_S$. The two sub functions are applied in the following order:

1. The *order function* $f_O$ produces a *pre-tag* $pL_m$ by ordering the semantics *A* found in $C_m$, in a total lexicographic order $(<^d)$. For example a subsequence

$C_m = \langle (c),(b),(a),(a),(c),(a) \rangle$ (times are omitted) is transformed into the pre-tag is of the form $pL_m = \langle (a),(a),(a),(b),(c),(c) \rangle$.

2. The *set function* $f_S$ operates on the semantics of the pre-tag $pL_m$ by eliminating the replicates and producing the tag $L_m$, where each type of semantic that appears in the subsequence appears only once in the produced tag. Continuing the previous example, for the pre-tag $pL_m = \langle (a),(a),(a),(b),(c),(c) \rangle$ the set function produces $L_m = \langle a,b,c \rangle$.

With both operations completed a subsequence of the form $C_m = \langle (c,t_1),(b,t_2),(a,t_3),(a,t_4),(c,t_5),(a,t_6) \rangle$ results to a tag of the form $(\langle a,b,c \rangle, t_1)$ (see Figure 5-2 for an example of an error tag).

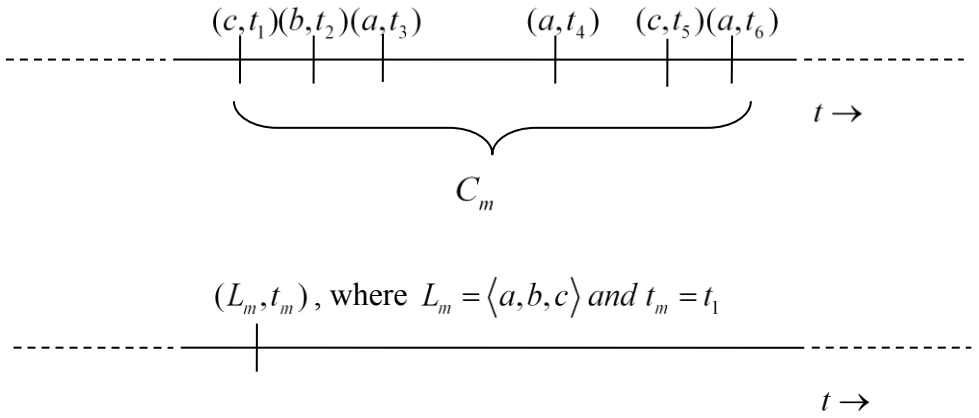



**Figure 5-2: From subsequence $C_m$ to tag $(L_m, t_m)$**

Once the tagging operation is complete, the collection of subsequences $C^R$ becomes a collection of tags $\bar{C}^R = \{(L_1,t_1),(L_2,t_2),(L_3,t_3),...,(L_M,t_M)\}$. Tags provide a better basis for the matching operation because much of the variation that is found in the subsequences has been reduced. For the remainder of this chapter the term *tag* we will be referring to the semantic information $L_m$ only.

## 5.2 Matching tags

### 5.2.1 Tag similarity based on the system's design

Professionals systems are large complex machines capable of performing a wide range of operations. To manage their complexity systems are built out of subsystems. Each subsystem is designed to provide certain functionality. Subsystems consist of components, which work together to provide that functionality. Components are interacting with each other passing signals or information in order to execute their operation in a coordinated manner. Sub-systems interact with each other via interfaces to deliver the overall system functionality. A good system engineering principle requires *strong coherence between components of the same subsystem and weak cohesion between sub-systems*. That means that components that belong to the same sub system are closely interconnected and dependent on each other when operating to provide a certain set of functionality. Components of different sub

systems have no direct connection to each other. They communicate via interfaces on sub-system level. This engineering configuration is known as modular system design. It allows sub systems to be developed more independently from each other but also enables them to sustain a level of resilience during operation against errors that occur in other sub systems. When an error occurs in a component it will very likely propagate to other components of the same subsystem because of the strong dependencies between the components. However other subsystem can withstand this error because their functionality is self-contained and there are mechanisms in place that monitor the interfaces and can handle "external" unexpected behavior.

The traces resulting from failures and recoveries reflect these design characteristic. This was verified by the fault injection experiment, where faults injected into a sub system produced subsequences that contained traces created by components that belong to that sub system. On the other hand, faults that are injected into different sub systems produce subsequences which rarely contain shared types of traces. These findings support the belief that there is a relation between the semantic content of subsequences and the modular design of the system. This relationship is also supported by the use of trace based fault signatures for fault diagnosis [Iye86]. Fault signatures are manifestations of the symptoms that the components show when the same errors occur. Fault signatures are possible because given the same error, most likely the same components will exhibit the same symptoms, a behavior that is based on the functional dependencies between these components.

The manifestation of symptoms is not happening in deterministic manner for each occurrence of the same error. The fault injection experiments (see 3.2) showed that even under controlled conditions, subsequences resulting from the same injected faults can vary in their semantic content. Variation can be found in the number of traces as well as the number of different types of traces (see 3.2.4). Moreover, given that the structure of subsequences can be affected by environmental factors acting on the logging mechanism, it is natural to expect that under operational conditions, variation in the traces of subsequences will be present. It is more realistic to expect that multiple occurrences of a particular type of physical error are represented by *similar* instances of subsequences rather than identical subsequences.

Which subsequences are similar enough to be perceived as originating from the same physical error is a matter of the definition of the similarity measure. In this thesis, it is assumed that systems are designed in a modular way. Therefore, when traces are frequently appearing together it is an indication that they originate from components that are functionally dependent. The stronger the functional dependency the more frequent the co-occurrence of traces is and the more similar these traces are perceived to be. If, for example, two components are directly dependent to each other, when one experiences an error the other *will* experience an error. If both components can log error traces, these traces will always appear together.

A similarity measure is required that can reflect the modular design of the system. The similarity between subsequences can then be evaluated on the basis of the manifestation of the *strong coherence /weak coupling principle of modular system design* in traces. The benefit of such a similarity measure is that the similarity can be evaluated directly from the sequence without using any system specific information.

The following subsections are organized as follow: In 5.2.2 an end to end matching process for traces is described. In 5.2.3 the cost function is defined, which is the main contribution of this chapter. An example of how the tag matching performs is given in 5.2.4.

## 5.2.2 Matching operation

The objective of the matching process is to group a given collection of tags obtained from a single system, into *tag types* according to their similarity. A tag type can contain multiple instances of tags. Though the discussion on comparing tags was based on the use of term "similarity", the grouping of tags into tag type is using as a measure the *dissimilarity* or *distance* between them.

Dissimilarity between two tags is measured with the help of the *edit distance.* The edit distance is defined by the cost of the *edit operations* that are performed on the semantics of one tag to transform it into the tag that to which it is compared with [The09]. The cost of each edit operation is defined by the *cost function.* In this section a cost function is defined that reflects the (dis)similarities between traces.

The tag matching operation is essentially a clustering operation. To perform the clustering the following elements are needed:

a. The clustering algorithm
b. The *cost function* which sets the cost of each edit operation. The cost function is nested in the edit-distance function.
c. An *edit-distance function*, computes the dissimilarity between two *tags*
d. A *stopping rule*, which helps to find the best clustering result given a collection of tags and the edit distance between them

In the following sections these four elements will be presented. First the context of the tag matching operation will be set by presenting the clustering algorithm, the stopping rule and the edit function in the sections 5.2.2.1-5.2.2.3. The first three sections also outline some of the requirements that are put on the cost function. Then in section 5.2.3.3 the cost function will be presented.

### 5.2.2.1 Clustering algorithm

Clustering tags into tag types is done on the basis of tag similarity. Tags that are very similar should be clustered into the same tag type. This type of clustering results to compact clusters. The agglomerative *complete link algorithm* is most suitable clustering algorithm for retrieving compact clusters from a data set [The06]. The agglomerative complete link clustering algorithm performs the clustering process bottom-up. It does that by iterating as many times as the number of tags in the initial collection of tags $LC_1$. At the starting position the agglomerative algorithm treats each object as a singleton cluster (a cluster containing one *tag*) then successively merges pairs of clusters until all clusters are merged into one cluster. Figure 5-3 shows an example of how the gradual clustering progresses from multiple clusters at the bottom (x-axis) to one final cluster containing all tags at the top of the dendrogram. The y-axis shows the distances where the clustering is taking place. In the first iteration the algorithm begins with as many clusters as tags (singleton clusters). As the algorithm iterates, one or more clusters are merged (branches in dendrogram merging) to form

single clusters until the final iteration where the last two remaining clusters are merged.

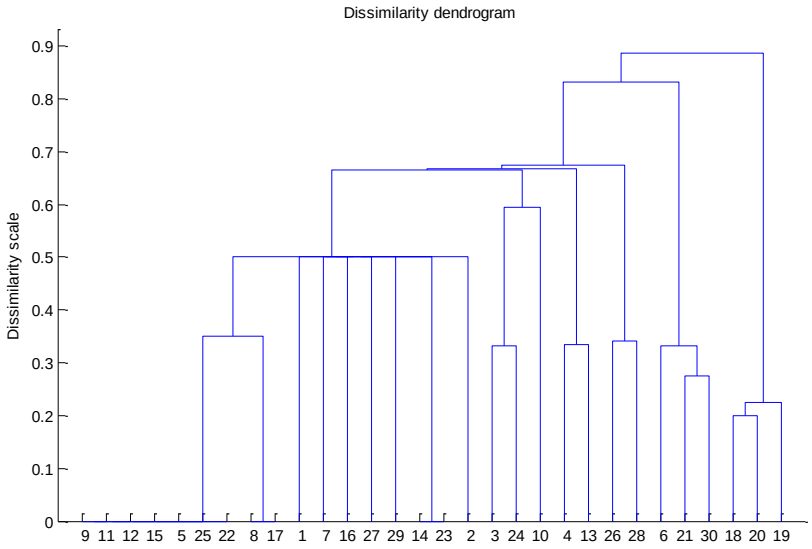



**Figure 5-3 Complete link clustering algorithm dendrogram**

The similarity between two clusters is computed based on their most dissimilar (distant) members. This computation rule serves the creation of *tag types* where the evidence on inter cluster similarity is strong to support their grouping.

To perform this operation, the algorithm requires as input the edit distances between the tags in the collection $LC_1$. The edit distances are provided in the form of a distance matrix $H_1$ that contains the pairwise dissimilarities between the tags in $LC_1$. At the start the distance matrix $H_1$ is a $M \times M$ matrix containing the edit distances between all pairs of tags in $LC_1$, where $LC_1 = \bar{C}^R$ and $\bar{C}^R = \{L_1, L_2, ..., L_M\}$. At every iteration of the algorithm, a new clustering result $LC_q$ is produced and the distance matrix $H_q$, where $q = \{1,2,3, \dots, M\}$, is recomputed.

More details on how the complete link clustering algorithm operates can be found in 9.5.3Appendix D

### 5.2.2.2 Stopping rule

For every iteration of the clustering algorithm a cluster result $LC_q$, $q = \{1,2,3, \dots, M\}$, is produced. To choose a clustering solution $LC_q^S$ that returns compact clusters of tags, a *stopping* or *cutting* rule is used. Literature is providing an abundance of stopping rules [Mil85] [Moj75] out of which the measure of *silhouettes* [Rou87] is chosen because it fits to ratio scale measures of dissimilarity and it favors compact and well separated clusters (tag types with very similar tags as members). The silhouette value measures how similar the tag is to tags of the same cluster compared to tags in other clusters, and it ranges from -1 to +1.

The silhouette value $s(L_m)$ for the $m^{\text{th}}$ tag is defined:

$$s(L_m) = \frac{b(L_m) - a(L_m)}{\max\{a(L_m), b(L_m)\}} \qquad \textbf{( 5-1)}$$

where $a(L_m)$ is the average distance from the $m^{th}$ tag to the other tags of the same cluster as $m$, and $b(L_m)$ is the minimum average distance from the $m^{th}$ tag to tag in a different cluster, minimized over clusters.

The average silhouette measure **$Sil_q$** for the qth clustering result $LC_q$ is defined as

$$Sil_q = \frac{\sum_{1}^{M} s(L_m)}{M} \qquad \textbf{( 5-2),}$$

where $m, q = \{1,2,3,\ldots,M\}$. The average silhouette measure can take values between $-1 \le Sil_q \le 1$. The higher the value of $Sil_q$, the more compact the clusters are in the clustering result.

#### 5.2.2.3 Edit-Distance function - metric properties

To initiate the clustering process the dissimilarity matrix $H_1$ is provided. The dissimilarity matrix $H_1$ contains the distance between any pair of singleton clusters in the tag collection $LC_1$. The distance between two singleton clusters is provided by the edit distance function. The distance functions needs to meet certain conditions that cascade to the edit distance function. Any distance function $F$ has to satisfy three conditions:

1. $F(L_i, L_j) \ge 0$, that of positivity, with equality only if $L_i = L_j$
2. $F(L_i, L_j) = F(L_j, L_i)$, that of symmetry
3. $F(L_i, L_j) \le F(L_i, L_k) + F(L_k, L_j)$,that of triangle inequality

Among the three, the triangle inequality is the most difficult to satisfy. Triangle inequality assures that when comparing objects, the distance between any two objects $F(B,C)$ is always the shortest way to go from $A$ to $B$ and no other way e.g. $F(B,A) + F(A,C)$ can be shorter (Figure 5-4).

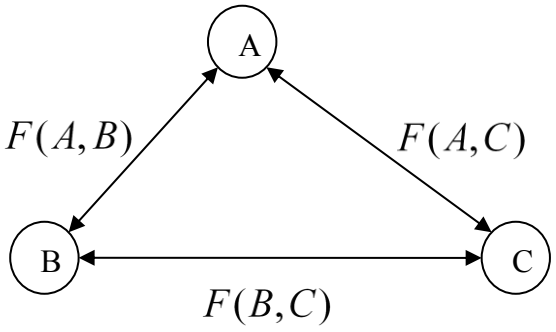



**Figure 5-4 Triangle inequality**

If the triangle inequality is not satisfied by a distance function at all times, a comparison of the distances between three objects can fail, compromising the performance of the clustering operation. The triangle inequality criterion is a criterion of good performance for the distance function but it is a *necessary* condition for the *cost function* as will be shown in 5.2.3.

The performance of a distance function can be measured with the ***inequality looseness test***. The test measures the proportion of tag triplets $(L_1, L_2, L_3)$ that belong

in a collection of tags $LC_1 = \bar{C}^R$ which *violate* the triangle inequality criterion [Vid88].

$$F(A,B,C) = F(A,B) + F(B,C) - F(A,C) \geq 0 \qquad \textbf{( 5-3)}$$

When the failure rate of the *inequality looseness test* is zero the triangular inequality requirement is not violated. The greater the failure rate is the more the validity of the comparison is compromised. *Low* failure rates do not compromise the validity of the comparison [Yuj07] [Mar93] [Vid88][Ars00].

#### 5.2.2.4 Edit distance function

A tag is a set of lexicographically ordered semantics $ic_i$, obtained from a finite pool of semantics $IC = \{ic_1, ic_2, ..., ic_w\}$. The finite pool of semantics $IC$ is similar to the letters of the alphabet and tags are like words that are formed by these letters. Given that, the distance functions that are appropriate for measuring the distance between tags are based on edit operations of string matching applications [Coh03].

The Levenshtein edit distance (LED) is a method that has been widely used in string matching applications and is known for its effectiveness and simplicity. The LED is estimating the dissimilarity between two tags $L_1$ and $L_2$, by performing the transformation from $L_1 \rightarrow L_2$[Lev66]. This transformation can be completed by a sequence of edit operations. There are three types of edit operations:

- Replacement $e(a \rightarrow b)$: a semantic $a$ in $L_1$ is replaced by another semantic $b$
- Deletion $e(a \rightarrow \lambda)$: a semantic $a$ in $L_1$ is deleted
- Insertion $e(\lambda \rightarrow b)$: a semantic $b$ in $L_1$ is inserted

There is *cost* associated with each edit operation of replacement, deletion and insertion respectively denoted by $\gamma(a \rightarrow b), \gamma(a \rightarrow \lambda), \gamma(\lambda \rightarrow a)$ respectively.

There are multiple combinations of edit operations that can transform a tag $L_1$into $L_2$. Any sequence of operations that completes the transformation is an editing *path* $P_e(L_1, L_2)$. The editing path for the transformation $L_1 \rightarrow L_2$ has a length of $Len(P_e)$, where $\max\ i, j\ \leq Len\ P_e\ \leq i + j$, which is defined as the number of elementary edit operations described by $P_e(L_1, L_2)$ [Mar93]. The total cost of an editing path is given by the sum of the cost of each edit operation in the path.

$$D(P_e(L_1, L_2)) = \sum_{k=1}^{Len(P_e)} \gamma(L_{i_{k-1}+1 \ldots i_k} \rightarrow L_{j_{k-1}+1 \ldots j_k}) \qquad \textbf{( 5-4),}$$

For (5.2-1) to hold the elementary cost function $\gamma$ has to be a metric [Mar93]

The edit distance $D_L(L_i, L_j)$ between two tags $L_i$ and $L_j$, is given by the *minimum cost* of transforming tag $L_i$ into $L_j$:

$$D_L(L_i, L_j) = \min(D(P(L_i, L_j)) \mid P(L_i, L_j)\ is\ an\ edit\ path\ from\ L_i\ to\ L_j) \qquad \textbf{( 5-5)}$$

The distance between any pair of tags $L_i$ and $L_j$ can be computed in this way. LED does not take into account the length of the tags.

One typical characteristic of tags is that their length can vary. Differences in the length of the tags can be found even if the tags represent the same type of physical event. Tags may differ in length by one or more semantics $ic$. The variation in the length of tags poses a problem for LED. It is more likely to find differences when comparing long tags than when comparing short tags. Consequently long tags have a disadvantage in finding matches compared to short tags when using LED. The comparison of tags has to be performed taking into account their lengths. Differences found in longer tags should be less important than difference in short tags. To compensate for that, the distance function has to take into account the length of the tag when computing the edit distance.

The shortcomings of LED regarding the consideration of tag lengths are overcome with the use of the Normalized Edit Distance (NED). The normalized edit distance is appropriate for tags of different lengths because it normalizes the cost of an edit transformation by the number of edit operations. In the literature there are two main definitions for a normalized edit distance.

The first normalized edit distance (NED1) is described by Marzal and Vidal [Mar93], where normalization is taking place on the minimum found for each edit length $Len(P_e)$. The normalized edit distance $nD_A(L_1, L_2)$ between two labels $L_1$ and $L_2$ is the minimum of the normalized distances of all edit lengths:

$$NED_1(L_1, L_2) = min\left(\frac{D(P_e(L_1,L_2))}{Len(P_e)}\right) \qquad (5\text{-}6)$$

In [Mar93] it is argued that when the global minimized edit distance $D_L(L_i, L_j)$ is found first and then it is normalized by the length of its edit path $Len(P_e)$ this gives wrong results.

Another definition of a normalized edit distance (NED2) is given by Yujian [Yuj07], which given the distance by LED between two labels, $D_L(L_1, L_2)$ and the length of those labels $|L_1|$ and $|L_2|$ the normalized edit distance is given by:

$$\mathbf{NED_2(L_1, L_2)} = \frac{\mathbf{2D_L(L_1,L_2)}}{\mathbf{q\ L_1 + L_2 + D_L(L_1,L_2)}} \qquad (5\text{-}7),$$

if $\forall\, \mathbf{a} \in \mathbf{IC}, \boldsymbol{\gamma}\ \boldsymbol{\lambda} \rightarrow \mathbf{a} = \boldsymbol{\gamma}\ \mathbf{a} \rightarrow \boldsymbol{\lambda} = \mathbf{q}$ and γ is a metric.

Method NED1 does not always meet the triangular inequality criterion of a metric depending on the data set [Mar93]. The failure rates for meeting the criterion, given a data set, can be computed using the inequality looseness test [Yuj07]. Method NED2 is always a metric if the conditions for γ are met [Yuj07]..

### 5.2.3 Cost function for traces

The cost for elementary edit operations is defined by the *cost function* $\gamma$. There are two basic requirements that the cost function needs to meet:

1. Represent the functional association of components
   Tags are grouped into tag types based on the assumption that there is an underlying functional association of the components that produced them. Coupled

occurrences of semantics in tags represent such associations. Therefore the cost function has to reflect the associations between semantics.

2. Meet all metric properties

#### 5.2.3.1 Association between semantics

To meet the first requirement the cost function needs to be capable of reflecting the association of traces as found in the sequence. The Jaccard coefficient that was used in section 3.3.3 can provide that measure because it shifts the weight for measuring the association on the co-occurrences of events and it ignores completely the number of occasions where both events are absent. By ignoring the occurrence of other events, this definition of association provides an independent association measure between the two events of interest:

$$\phi_{AB} = \frac{x_{11}}{x_{11} + x_{12} + x_{21}} \qquad \textbf{( 5-8)}$$

As a reminder to the reader:

$x_{11}$: attributes $A$ and $B$ are both present

$x_{12}$: attribute $A$ is present but $B$ is absent

$x_{21}$: attribute $B$ is present but $A$ is absent

The coefficient takes values $0 \le \phi \le 1$, where $\phi = 0$ indicates independence (no similarity) between $A$ and $B$, and $\phi = 1$ when $A$ and $B$ are absolutely dependent to each other (identical). The coefficient is symmetric, i.e. $\phi_{ij} = \phi_{ji}$.

However the coefficient is used here differently than in 3.3.3. Instead of measuring the occurrence of traces in arbitrary segments of the sequence, here the measurement is done on the occurrence of semantics within the tags. The degree of association between *ic*s is measured as they are found in the tags of the collection $LC_1 = \bar{C}^R$. The association coefficient is computed for all pairs of *ic*s in $IC = \{ic_1, ic_2, ..., ic_w\}$ that are found in $LC_1$.

#### 5.2.3.2 Computation of $\phi$ based on observations in labels sets

Given $\bar{C}^R$, the collection of tags produced by the tagging process, the input set for the clustering is $LC_1 = \bar{C}^R$ . The set $LC_1$ contains *M* tags. Every tag contains a collection of semantics *ic*s. The association coefficient $\phi_{ij}$ is computed for all the semantics $|IC| = w$ that are found in the tags of $LC_1$ and it expresses the association between semantic $ic_i$ and $ic_j$. The values of $\phi_{ij}$ are arranged in a $w \times w$ matrix $\Phi$ where the rows and columns are ordered lexicographically $ic_1 <^d ic_2 <^d ic_3, ..., <^d ic_w$ (see Table 5-1).

| | $ic_1$ | $ic_2$ | ... | $ic_w$ |
|---|---|---|---|---|
| $ic_1$ | $\phi_{11}$ | $\phi_{12}$ | ... | $\phi_{1w}$ |
| $ic_2$ | $\phi_{12}$ | $\phi_{22}$ | ... | $\phi_{2w}$ |
| ⋮ | ⋮ | ⋮ | ⋱ | ⋮ |
| $ic_w$ | $\phi_{w1}$ | $\phi_{w2}$ | ... | $\phi_{ww}$ |

**Table 5-1 $w \times w$ Matrix $\Phi$ of pair wise association coefficients between semantics**

To test whether the association between $ic_i$ and $ic_j$ is significantly different than that of a random association, at least 20 observations are required [Gri67]. A test statistic $X_\phi^2$ is used to perform the test. The test statistic $X_\phi^2$ is defined as:

$$X_\phi^2 = \frac{(x_{11}x_{22} - x_{21}x_{12})^2 n}{(x_{11}+x_{22})(x_{11}+x_{21})(x_{21}+x_{22})(x_{12}+x_{22})} \qquad (5\text{-}9),$$

$x_{22}$ : attribute $B$ and $A$ are both absent

The test statistic $X^2$ follows then approximately a $\chi^2$ distribution with 1 degree of freedom [Gri67].

#### 5.2.3.3 Cost of edit operations

The cost function $\gamma$ is described by a $(w+1)\times(w+1)$ matrix. The $w \times w$ elements of the body of the matrix contain the cost for the pairwise replacement operations $ic_i \to ic_j$, the bottom row $(w+1)$ contains the cost of the insertion operations $\lambda \to ic_i$ and the right most column $(w+1)$ contains the cost of the deletion operations $ic_i \to \lambda$

.

| | $ic_1$ | $ic_2$ | ... | $ic_w$ | $\lambda$ |
|---|---|---|---|---|---|
| $ic_1$ | $\gamma(ic_1 \to ic_2)$ | $\gamma(ic_1 \to ic_2)$ | ... | $\gamma(ic_1 \to ic_w)$ | $\gamma(\lambda \to ic_1)$ |
| $ic_2$ | $\gamma(ic_2 \to ic_1)$ | $\gamma(ic_2 \to ic_2)$ | ... | $\gamma(ic_2 \to ic_w)$ | $\gamma(\lambda \to ic_2)$ |
| ⋮ | ⋮ | ⋮ | ⋱ | ⋮ | ⋮ |
| $ic_w$ | $\gamma(ic_w \to ic_1)$ | $\gamma(ic_w \to ic_2)$ | ... | $\gamma(ic_w \to ic_w)$ | $\gamma(\lambda \to ic_w)$ |
| $\lambda$ | $\gamma(ic_1 \to \lambda)$ | $\gamma(ic_2 \to \lambda)$ | ... | $\gamma(ic_w \to \lambda)$ | $\gamma(\lambda \to \lambda)$ |

**Table 5-2 Cost function $\gamma$**

#### 5.2.3.4 Cost of replacement operations

Given the association matrix $\Phi$ for a set $LC_1$, the cost of the replacement operations is defined as:

$$\gamma(ic_i \to ic_j) = 1 - \phi_{ij} \quad \textbf{(5-10)}$$

The measure takes values in $0 \le \gamma(ic_i \to ic_j) \le 1$ **(5-11)**

Intuitively the cost function suggests that strongly associated *ic*s (high values of $\phi_{ij}$) are economical to replace (low values of $\gamma(ic_i \to ic_j)$) because their strong association suggests strong similarity. The opposite applies for weakly associated *ic*s.

Given the above definition the weights of replacement satisfy all conditions of a metric:

**Positivity**: $0 \le \gamma(ic_i \to ic_j) \le 1$, derives directly from the definition of the coefficient and (5.2-5)

**Symmetry**: $\gamma(ic_i \to ic_j) = \gamma(ic_j \to ic_i)$, $\phi_{ij} = \phi_{ji}$

**Triangular inequality**: for triplet $(ic_i, ic_j, ic_k)$,

$\gamma(ic_i \to ic_k) \le \gamma(ic_i \to ic_j) + \gamma(ic_j \to ic_k)$ (the proof is lengthy and is therefore not included here, further reading in [Gow86], Theorem 10, page 15)

#### 5.2.3.5 Cost function for deletion and insertion operations

For the edit operations of insertion and deletion, the coefficient of association measure does not satisfy conceptually as it does for the replacement operation. If an *ic* needs to be inserted or deleted the cost of the operation has to have a global interpretation rather than its association with another *ic*. Here a different approach is followed. The edit operations of insertion and deletion of *ic*s have a weigh that depends on the *abundance* or *rarity* of an *ic* in the preprocessed *sequence* i.e. after the removal of ppt. The cost of deletion and insertion are defined as follows:

The relative frequency of $ic_i$ in the sequence is given by

$rf_i = \dfrac{f_i}{\sum_{k=1}^{w} f_k}$, where $f_i$ is the frequency of $ic_i$ and $\sum_{k=1}^{w} f_k$ the sum of frequencies of all *ic*s

The measure takes values $0 \le rf_i \le 1$ **(5-12)**

and the sum of all relative frequencies is $\sum_{k=1}^{w} rf_i = 1$ **(5-13)**

The weight of insertion and deletion of $ic_i$ is defined as:

$$\gamma(\lambda \to ic_i) = \gamma(ic_i \to \lambda) = 1 - rf_i \quad \textbf{(5-14)}$$

Intuitively the above definition suggests that rare *ic*s are more expensive to insert or to delete than frequent ones. The weights for insertion and deletion as defined here, satisfy all conditions of a metric:

**Positivity**: $0 \le \gamma(\lambda \to ic_j) \le 1$

**Symmetry**: $\gamma(\lambda \to ic_i) = \gamma(ic_i \to \lambda)$

**Triangular inequality**: $\gamma(ic_i \to ic_k) \le \gamma(ic_i \to \lambda) + \gamma(\lambda \to ic_k)$ does always hold since: $\gamma(ic_i \to \lambda) + \gamma(\lambda \to ic_k) = 1 - rf_i + 1 - rf_k = 2 - (rf_i + rf_k) \ge 1$

The cost function $\gamma$ as defined satisfies the criteria of a metric. This allows the use of the normalized edit distances NED. However for NED2 the conditions for γ are not met since $\gamma\ \lambda \to ic\ = \gamma\ ic \to \lambda\ \neq q, \forall\ ic$.

### 5.2.4 Tag matching example

An example is given to demonstrate the performance of the normalized edit distance versus Levenshtein edit distance. Also the performance of the cost function is examined. In the example ics are represented by lower case letters of the alphabet. The set of tags to be matched into tag types are shown in Table 5-3 Collection of tags.

| Index | Tag |
|---|---|
| 1 | <a,b,c> |
| 2 | <a,b> |
| 3 | <a,c> |
| 4 | <a,b,c,d> |
| 5 | <k,l,m,n> |
| 6 | <k> |
| 7 | <k,l> |
| 8 | <m,n> |
| 9 | <m,n,d> |
| 10 | <y,z> |

**Table 5-3 Collection of tags**

The tags 1, 2, 3 and 4 originate from the same module. The most extended form of the tag is <a,b,c> but because of the variation in the logging mechanism the tag appears also as <a,b> or <b,c> . The tag <a,b,c,d> results from the addition of the *ic* "d" to the tag <a,b,c>. The *ic* "d" however is not specific to this module. It is a common error trace that occurs frequently with different types of error events. The tags 5, 6, 7, 8, and 9 originate from the same module (second module). The tag <k,l,m,n> is the extended tag form of this error. Due to variation the tag rarely appears as <k,l> , <m,n> or <m,n,d>. The 10$^{th}$ tag <y,z> is originating from another module (third module) not related to any of the above.

The above tags would be represented correctly by 3 tag types according to their origin. First tag type contains tags 1,2,3,4, second tag type contains tags 6,7,8,9, and the third contains tag 10.

To perform the matching operation the cost function needs to be defined. The cost function is defined by the associations between *ic*s for the replacements operation and their relative frequencies in the sequence for the deletion and insertion operations.

The associations between *ic*s can be seen in Table 5-4

| | a | b | c | d | k | l | m | n | y | z |
|---|---|---|---|---|---|---|---|---|---|---|
| **a** | 1 | 0.6 | 0.6 | 0.3 | 0 | 0 | 0 | 0 | 0 | 0 |
| **b** | 0.6 | 1 | 0.6 | 0.3 | 0 | 0 | 0 | 0 | 0 | 0 |
| **c** | 0.6 | 0.6 | 1 | 0.3 | 0 | 0 | 0 | 0 | 0 | 0 |
| **d** | 0.3 | 0.3 | 0.3 | 1 | 0.3 | 0.3 | 0.3 | 0.3 | 0.3 | 0.3 |
| **k** | 0 | 0 | 0 | 0.3 | 1 | 1 | 0.9 | 0.9 | 0 | 0 |
| **l** | 0 | 0 | 0 | 0.3 | 1 | 1 | 0.9 | 0.9 | 0 | 0 |
| **m** | 0 | 0 | 0 | 0.3 | 0.9 | 0.9 | 1 | 1 | 0 | 0 |
| **n** | 0 | 0 | 0 | 0.3 | 0.9 | 0.9 | 1 | 1 | 0 | 0 |
| **y** | 0 | 0 | 0 | 0.3 | 0 | 0 | 0 | 0 | 1 | 1 |
| **z** | 0 | 0 | 0 | 0.3 | 0 | 0 | 0 | 0 | 1 | 1 |

**Table 5-4 Associations between *ic*s**

The associations between *ic*s are in line with their origin. The *ic*s always occur together is one e.g. $\varphi_{y,z} = 1$. The ics that often occur together have high association e.g. $\varphi_{b,c} = 0.6,\ \varphi_{k,m} = 0.9,\ \varphi_{l,n} = 0.9$ . The associations between *ic*s that never occur together are zero e.g. $\varphi_{a,k} = 0$.

The *ic* "d" is the most frequent trace in the preprocessed sequence occurring 50% of the time. The remaining 50% of occurrence is equally spread across the other *ic*s. The relative frequencies of the *ic*s as measured before the tagging operation are:

| a | b | c | d | k | l | m | n | y | z |
|---|---|---|---|---|---|---|---|---|---|
| 0.055 | 0.055 | 0.055 | 0.5 | 0.055 | 0.055 | 0.055 | 0.055 | 0.055 | 0.055 |

**Table 5-5 Relative frequencies of *ic*s**

From the associations and the relative frequencies the cost function is defined as follows:

| | a | b | C | d | k | l | m | n | y | z | $\lambda \rightarrow$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **a** | 0 | 0.4 | 0.4 | 0.7 | 1 | 1 | 1 | 1 | 1 | 1 | 0.945 |
| **b** | 0.4 | 0 | 0.4 | 0.7 | 1 | 1 | 1 | 1 | 1 | 1 | 0.945 |
| **c** | 0.4 | 0.4 | 0 | 0.7 | 1 | 1 | 1 | 1 | 1 | 1 | 0.945 |
| **d** | 0.7 | 0.7 | 0.7 | 0 | 0.7 | 0.7 | 0.7 | 0.7 | 0.7 | 0.7 | 0.5 |
| **k** | 1 | 1 | 1 | 0.7 | 0 | 0 | 0.1 | 0.1 | 1 | 1 | 0.945 |
| **l** | 1 | 1 | 1 | 0.7 | 0 | 0 | 0.1 | 0.1 | 1 | 1 | 0.945 |
| **m** | 1 | 1 | 1 | 0.7 | 0.1 | 0.1 | 0 | 0 | 1 | 1 | 0.945 |
| **n** | 1 | 1 | 1 | 0.7 | 0.1 | 0.1 | 0 | 0 | 1 | 1 | 0.945 |
| **y** | 1 | 1 | 1 | 0.7 | 1 | 1 | 1 | 1 | 0 | 0 | 0.945 |
| **z** | 1 | 1 | 1 | 0.7 | 1 | 1 | 1 | 1 | 0 | 0 | 0.945 |
| $\rightarrow \lambda$ | 0.945 | 0.945 | 0.945 | 0.5 | 0.945 | 0.945 | 0.945 | 0.945 | 0.945 | 0.945 | Inf |

**Table 5-6 Cost function for edit operations**

From the cost function it can be seen that when *ic*s are strongly associated their cost of replacement is low and vice versa. When an *ic* is abundant in the sequence it is also cheaper to delete or insert than a rarely occurring *ic*. With the cost function defined the distance between the tags can be calculated. These distances are used the clustering of the tags into tag types using the complete link clustering algorithm.

### 5.2.4.1 Matching tags

To demonstrate the benefits of the normalized edit distance over the non-normalized, bot algorithms are used and the results are compared.

#### *5.2.4.1.1 Levenshtein edit distance (non-normalized)*

The edit distances computed by the Levenshtein algorithm are shown in Table 5-7.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 0 | 0.945 | 0.945 | 0.5 | 3.945 | 2.89 | 2.945 | 2.945 | 2.7 | 2.945 |
| **2** | 0.945 | 0 | 0.4 | 1.445 | 3.89 | 1.945 | 2 | 2 | 2.5 | 2 |
| **3** | 0.945 | 0.4 | 0 | 1.445 | 3.89 | 1.945 | 2 | 2 | 2.5 | 2 |
| **4** | 0.5 | 1.445 | 1.445 | 0 | 3.7 | 3.39 | 3.445 | 3.445 | 2.945 | 3.445 |
| **5** | 3.945 | 3.89 | 3.89 | 3.7 | 0 | 2.835 | 1.89 | 1.89 | 1.745 | 3.89 |
| **6** | 2.89 | 1.945 | 1.945 | 3.39 | 2.835 | 0 | 0.945 | 1.045 | 1.545 | 1.945 |
| **7** | 2.945 | 2 | 2 | 3.445 | 1.89 | 0.945 | 0 | 0.2 | 0.7 | 2 |
| **8** | 2.945 | 2 | 2 | 3.445 | 1.89 | 1.045 | 0.2 | 0 | 0.5 | 2 |
| **9** | 2.7 | 2.5 | 2.5 | 2.945 | 1.745 | 1.545 | 0.7 | 0.5 | 0 | 2.5 |
| **10** | 2.945 | 2 | 2 | 3.445 | 3.89 | 1.945 | 2 | 2 | 2.5 | 0 |

**Table 5-7 Edit distances as computed with the Levenshtein edit distance algorithm**

Using the edit distance in Table 5-7, the tags are clustered into tag types using the agglomerative algorithm. The clustering of tag is an iterative process. The sequence with which the tags are clustered is represented by the dendrogram of Figure 5-5. The dendrogram is read bottom-up. The clustering process begins with singleton clusters i.e. each cluster contains one member. The dendrogram shows which clusters (x -axis) are clustered together and at what distance (y-axis). At each clustering iteration the clustering of tags is represented by the merging of their branches. A clustering result is obtained where the dendrogram is cut horizontally.

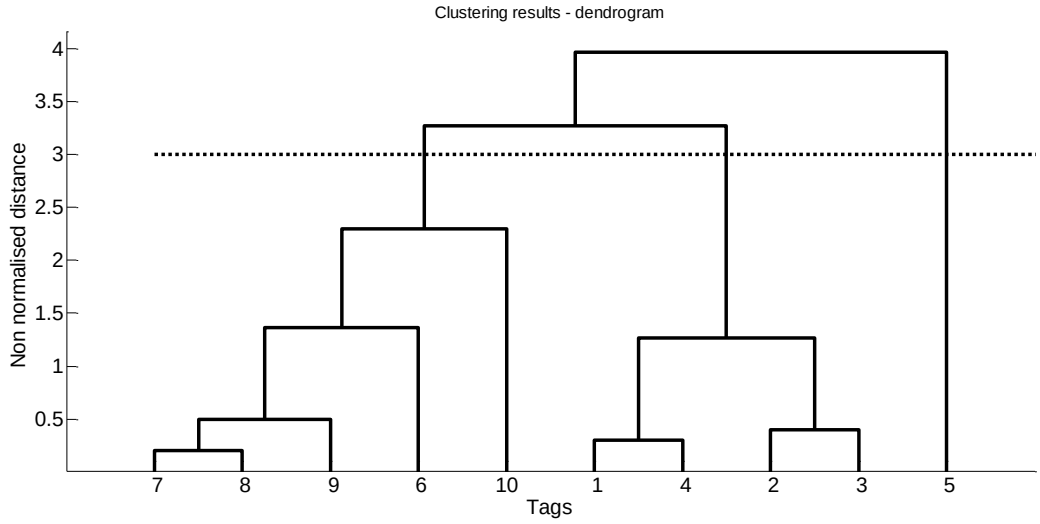



**Figure 5-5 Dendrogram of clustering results using non-normalized edit distances**

The horizontal dotted line in Figure 5-5 shows when the clustering algorithm has clustered the tags into 3 tag types. The result is not the expected outcome because tag 10 has been clustered together with tag 6,7,8,9. The distance of tag 10 from tags 6, 7, 8and 9 is short because of the tags short length. On the other and tag 5 is not grouped together with 6, 7, 8 and 9. Indeed tag 5 is added to the cluster only in the last iteration of the clustering process and after tag 1,2,3,4 are clustered together with 6, 7,

8, 9, and 10. Tag 5 is the furthest away from any other tag in the set because it is a lengthy tag.

#### *5.2.4.1.2 Normalized edit distance*

In the table below the normalized edit distances between the tags in the set can be seen.

| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 0 | 0.315 | 0.315 | 0.125 | 0.945 | 0.945 | 0.945 | 0.945 | 0.8613 | 0.945 |
| **2** | 0.315 | 0 | 0.2 | 0.3613 | 0.945 | 0.945 | 0.945 | 0.945 | 0.8333 | 0.945 |
| **3** | 0.315 | 0.2 | 0 | 0.3613 | 0.945 | 0.945 | 0.945 | 0.945 | 0.8333 | 0.945 |
| **4** | 0.125 | 0.3613 | 0.3613 | 0 | 0.889 | 0.8475 | 0.8613 | 0.8613 | 0.7363 | 0.8613 |
| **5** | 0.945 | 0.945 | 0.945 | 0.889 | 0 | 0.7088 | 0.4725 | 0.4725 | 0.4363 | 0.945 |
| **6** | 0.945 | 0.945 | 0.945 | 0.8475 | 0.7088 | 0 | 0.4725 | 0.5225 | 0.515 | 0.945 |
| **7** | 0.945 | 0.945 | 0.945 | 0.8613 | 0.4725 | 0.4725 | 0 | 0.1 | 0.2333 | 0.945 |
| **8** | 0.945 | 0.945 | 0.945 | 0.8613 | 0.4725 | 0.5225 | 0.1 | 0 | 0.1667 | 0.945 |
| **9** | 0.8613 | 0.8333 | 0.8333 | 0.7363 | 0.4363 | 0.515 | 0.2333 | 0.1667 | 0 | 0.8333 |
| **10** | 0.945 | 0.945 | 0.945 | 0.8613 | 0.945 | 0.945 | 0.945 | 0.945 | 0.8333 | 0 |

**Table 5-8 Edit distance computed with the normalized edit distance**

In the Table 5-8 it can be seen how the distances between tags are on a scale from 0 to 1. The dendrogram of the tag clustering process using the normalized edit distances is shown in Figure 5-6.

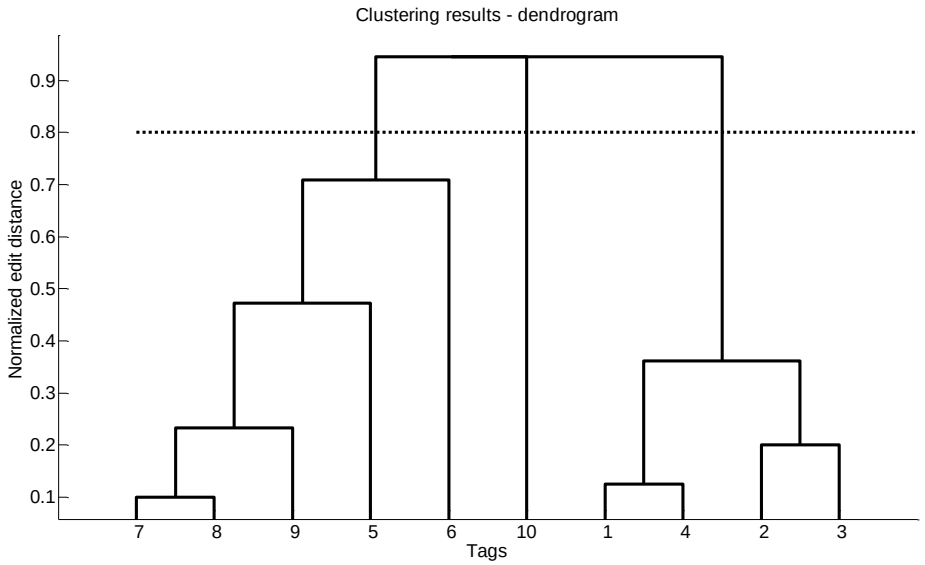



**Figure 5-6 Dendrogram of clustering results using the normalized edit distances**

When the clustering is terminated at 3tag types, the tags are clustered as expected. One tag type contains tags 1, 2, 3, 4, another contains 5, 6, 7, 8, 9 and a third tag type contains tag 10. Because of the normalization of the edit distance, tag 5 though it is lengthy, it is still similar to tags 7, 8, 9, and 6. Also, tag 10 though short in length is now distant from all the nonrelated tags.

#### *5.2.4.1.3 The cost function and edit path defines the normalized edit distance*

The example is continued with the normalized edit distance only. Few cases are picked out of the tag matching process to demonstrate better how the cost function and the edit paths define the edit distances.

**Case 1:** Given the same length of edit path and cost of edit operations the edit distance between tags is the same. The distance between tags 1 and 2 is equal to the distance between tags 1 and 3.

$$tag\ 1 = abc => a \rightarrow a => abc => b \rightarrow b => abc => c \rightarrow \lambda => ab = tag\ 2$$
$$D\ P_e\ tag1, tag2 = 0.945$$
$$L\ P_e = 3$$
$$NED\ tag1, tag2 = \frac{0.945}{3} = 0.315$$

$$tag\ 1 = abc => a \rightarrow a => abc => b \rightarrow \lambda => a\_c => c \rightarrow c => ac = tag\ 3$$
$$D\ P_e\ tag1, tag3 = 0.945$$
$$L\ P_e = 3$$
$$NED\ tag1, tag2 = \frac{0.945}{3} = 0.315$$

**Case 2:** The distance between tag 5 and 9 is shorter than the distance between tag 5 and 7 (or 8) even though there are more different *ic*s between tags 5 and 9 than there are between tag 5 and 7. This is due to the ability of the algorithm to utilize all similarities of the *ic*s even if the *ic*s are not perfectly aligned. It also is due the low cost of deleting "d".

$$tag\ 5 = klmn => k \rightarrow m => mlmn => l \rightarrow \lambda => m\_mn => m \rightarrow n => m\_nn$$
$$=> n \rightarrow d => mnd = tag\ 9$$
$$D\ P_e\ tag5, tag9 = 1.745$$
$$L\ P_e = 4$$
$$NED\ tag5, tag9 = 0.4362$$

$$tag\ 5 = klmn => k \rightarrow k => klmn => l \rightarrow l => klmn => m \rightarrow \lambda => kl\_n =$$
$$> n \rightarrow \lambda => kl = tag\ 7$$
$$D\ P_e\ tag5, tag8 = 1.89$$
$$L\ P_e = 4$$
$$NED\ tag5, tag8 = 0.4725$$

**Case 3:** The cost of deletion and/or insertion can burden considerably the distance between two seemingly similar tags. The distance between tag 8 and tag 6 is larger than the distance between tag 8 and tag 7 even though tags 8 and 6 have one common ic and tags 8 and 7 have none.

$$tag\ 8 = mn => m \rightarrow k => km => m \rightarrow \lambda => k = tag\ 6$$
$$D\ P_e\ tag8, tag6 = 1.045$$
$$L\ P_e = 2$$
$$NED\ tag8, tag6 = 0.5225$$

$$tag\ 8 = mn => m \to k => km => m \to l => kl = \ tag\ 6$$
$$D\ \ P_e\ \ tag8, tag7\ \ \ = 0.2$$
$$L\ \ P_e\ \ = 2$$
$$NED\ \ tag8, tag7\ \ = 0.1$$

#### *5.2.4.1.4 Silhouette value as the clustering stopping rule*

To control the clustering process result, the silhouette value is used as a stopping rule. The objective is to obtain as fewer tag types as possible to achieve maximum data compression by defining tags types that are compact i.e. tags within the tag types are more similar to the other tags of the same tag type than the tag of other tag types. The average silhouette value for the clustering process using the normalized edit distance is shown in Figure 5-7. The clustering process returns 3 tag types at an average silhouette value $sil_3 = 0.6396$, suggesting that the tags are arranged in compact clusters. If the clustering process is stopped one step later the algorithm returns 1 tag types for all tags. This is because tag 10 has the same maximum distance from the tags 1,2,3,4 and 5,6,7,8,9 . The silhouette for a single tag type drops at $sil_1 = 0.3497$. If the clustering process is stopped one step earlier at 4 tag types, the average silhouette value is again lower at $sil_4 = 0.6386$. The drop of the average silhouette value is due to the removal of tag 6 from the tag type that contains tag 5, 7, 8 and 9. By removing tag 6 the silhouette value for tag 6 increases to 1 as it becomes a singleton cluster (see Table 5-9), but this results to the drop of the silhouette values for tags 5, 7, 8, and 9 leading to a drop in the average silhouette value for the entire result. If the clustering process is stopped at an earlier stage e.g. at 5 tag type, the average silhouette value is higher but the gain in data compression is reduced as more singleton clusters are being returned.

In general it is expected that the clustering process will be stopped at average silhouette values that are lower than the optimal value of 1. This is the case because the tags are not identical and the string matching operation is calculating distances between tags are different than 1 and 0.

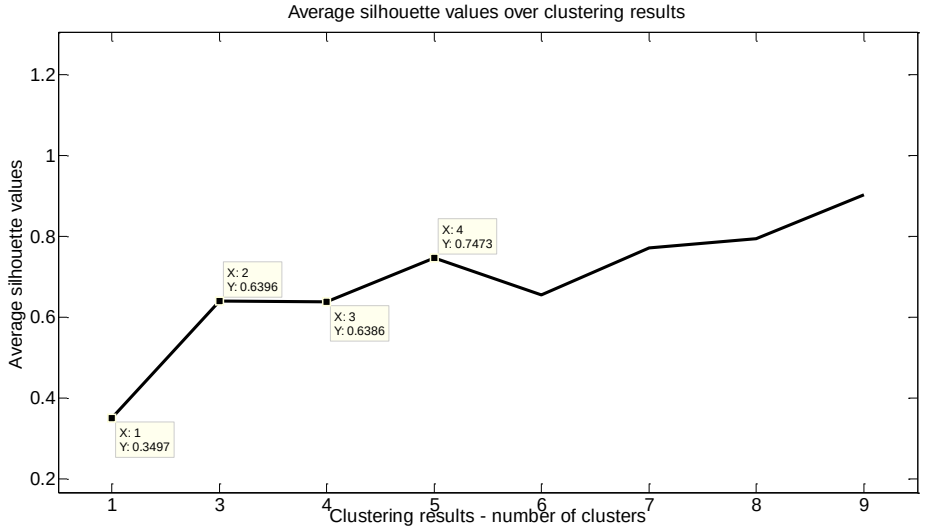



**Figure 5-7 Average silhouette value for clustering results**

| | Clustering results – number of clusters | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **1** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **Tag 1** | 0.2754 | 0.7003 | 0.699 | 0.6967 | 0.4237 | 0.4237 | 0.4237 | 1 |
| **Tag 2** | 0.2713 | 0.6784 | 0.6765 | 0.6733 | 0.4212 | 1 | 1 | 1 |
| **Tag 3** | 0.2713 | 0.6784 | 0.6765 | 0.6733 | 0.4212 | 1 | 1 | 1 |
| **Tag 4** | 0.2569 | 0.6365 | 0.6357 | 0.6284 | 0.4884 | 0.4884 | 0.4884 | 1 |
| **Tag 5** | 0.2291 | 0.4373 | 0.3466 | 1 | 1 | 1 | 1 | 1 |
| **Tag 6** | 0.2178 | 0.3967 | 1 | 1 | 1 | 1 | 1 | 1 |
| **Tag 7** | 0.3388 | 0.6497 | 0.4186 | 0.6279 | 0.6279 | 0.6279 | 0.5627 | 0.5627 |
| **Tag 8** | 0.3379 | 0.6475 | 0.5036 | 0.6751 | 0.6751 | 0.6751 | 0.4668 | 0.4668 |
| **Tag 9** | 0.2987 | 0.5713 | 0.4292 | 0.498 | 0.498 | 0.498 | 1 | 1 |
| **Tag 10** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Table 5-9 Silhouette values of each tag at each clustering result**

#### *5.2.4.1.5 Inequality Looseness test*

To test whether the all metric properties of the normalized edit distance are met, the inequality looseness test is performed on the values of normalized edit distance of Table 5-8. For the 10 tags in the example there are 120 triplets. The results of the test can be seen in Figure 5-8. There is no triplet that fails the test i.e. there is no outcome with a negative value. The normalized edit distance using the cost function of this example meets all metric properties.

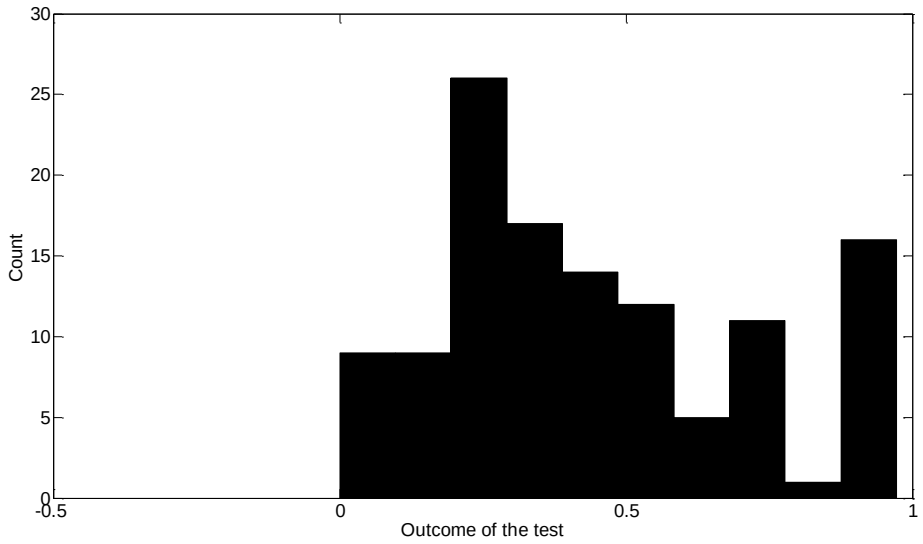



**Figure 5-8 Histogram of outcome of the inequality looseness test**

## *5.3 Discussion and Conclusions*

This chapter presented the last two steps of the transformation process. First step is the transformation of subsequences into tags. Tags are semantically simplified versions and point representations of the subsequences. With the tagging the data size is already reduced. The next step is the tag matching operation. This operation is technically more challenging. Tags are examined for their similarities and grouped into tag types. To perform comparison, a tag matching operation is defined that consists of several elements. Among those elements the newly defined *cost function* is specifically designed for being applied on traces and used by edit distance functions. The definition of the cost function remains true to the requirement not to relay on external or system specific information. It utilizes the semantic information found in traces and allows comparing tags for their similarities.

From the example in 5.2.4 it can be seen that clustering of tag into tag types depends greatly on the cost of deletion and insertion. This is the case because tags can vary in their length (the number of *ic*s in the tag). With the current definition of the cost function the deletion or insertion of an *ic* depends strictly on relative frequency of the *ic*s as this is measured before the tagging of the sequences. Another version of the cost function can be defined to take into account associations when deleting and inserting *ic*s so that when tags differ in length but do so with *ic*s that are associated an additional discount on the cost can be applied to reduce their distance between these tags.

The proposed cost function meets all metric properties and reflects the engineering relevance between semantics. The matching operation contributes further to the reduction of the data size of the sequence by grouping tag into tag types. Once complete a sequence is represented by a set of event type representatives rather than event instance representatives. This stage of the transformation will be presented in the case study of chapter 7.

# Chapter 6

# 6 Utilizing traces from multiple systems

In real life applications, multiple identical systems that are geographically distributed form a single observation group. Depending on the observed performance of these systems, decisions are made to act on each system individually e.g. corrective maintenance, or to all system collectively e.g. preventive maintenance. The transformation process described in chapters 4 and 5 is applied to each system individually. This allows the fitting of parameters for the segmentation (cutoff parameter) and the tag matching (cost function) suitable to the sequence generated by each system. This custom fitting approach requires however a "burn in" period for each system, where a sequence that will contain traces of error and recovery events has to be collected to be used for fitting the transformation tools, before the application can begin. Considering that professional systems are generally reliable products and failures occur rarely, this "burn in" period can be lengthy. Consequently the *transformed* format of the sequence for a newly installed system will not be available until the "burn in" period has elapsed.

These systems are identical and the structures of error and recovery subsequences of distributed systems should show similar forms. The similarity of subsequences across different systems can be described using the temporal and semantic structure of subsequences. The temporal structure of subsequences is capture by the cutoff parameter and the semantic structure is captured by the cost function. If the parameter values of several systems are similar, a representative value for each type of parameter can be used to characterize the group. The group characteristic parameter values can be used for the initialization of the transformation process of newly installed identical systems. For a newly installed system, the system group values can be used until enough traces are collected to make a custom fit of the transformation algorithms.

To follow such an approach, a degree of confidence is needed that the temporal and semantic structures of subsequences of multiple systems are coherent. This is particularly important since the algorithms are based on unsupervised learning techniques. The lack of coherent data structures among different systems poses a difficulty for the large scale applications of unsupervised machine learning tools[Pro00][Han00]. Incoherence of data structures can prohibit the applicability of a data mining methods from one type of system to another. Given that these groups consist of identical systems, which share identical design and logging mechanisms, coherence of the characteristics of subsequences is expected. However, even in the case of identical system it has been shown that the level and the type of the system workload have an effect on the manifestation of errors in the data sequences [Iye82]. In this thesis it is assumed that identical systems show similar patterns in the temporal and semantic structure of their subsequences unless there is proof of the opposite. In his chapter a method is presented to test the temporal and semantic characteristics of subsequences obtained from multiple systems, for evidence of non-coherence.

## 6.1 Characteristic subsequence structure of a system group

Given a group of systems where the parameterization of the segmentation algorithm and the cost function has been completed, the group characterizations can be made based on:

- For each system the segmentation cutoff parameter value represents the distance between successive traces that assigns traces in the sequence to the most compact clusters. Do identical systems have a characteristic value for the cutoff parameter?
- For each system the association between any two semantics represents the association between the components. Do identical systems have a characteristic value for the pairwise associations?

To accept that the group of systems produces subsequences of coherent structure characteristics, temporal and semantic, the values of the respective parameters have to demonstrate clear signs of localization in their distribution. Plotted as a histogram, such localization is manifested as a unimodal distribution.

Although tests for unimodality do exist [Har85] [Fis94] they tend to be quite elaborate requiring extensive simulations to estimate parameter values. In this thesis a simpler approach is followed. To determine whether the values of a characteristic have a unimodal empirical distribution the evidence is collected in two simple steps:

1. Visual inspection of the data that will identify whether the histogram is unimodal
2. A formal hypothesis test to support the belief that what appears to be a unimodal distribution is not a uniform distribution.

### 6.1.1 Visual Inspection

A visual inspection of the histogram of the values of the characteristic of interest i.e. cutoff values and pair wise associations between semantics can reveal directly whether a single mode exists. The location of the single mode of the empirical distribution is not very restrictive although there are some preferences depending on the parameter type. For the cutoff parameter for example the empirical distribution should ideally have a mode at the low end of range because this indicates compact subsequences through the group (short distance between successive traces). For pair wise associations the mode ideally should demonstrate either strong (high end of range) or weak (low end of range) associations. In Figure 6-1 two examples of unimodal histograms are shown.

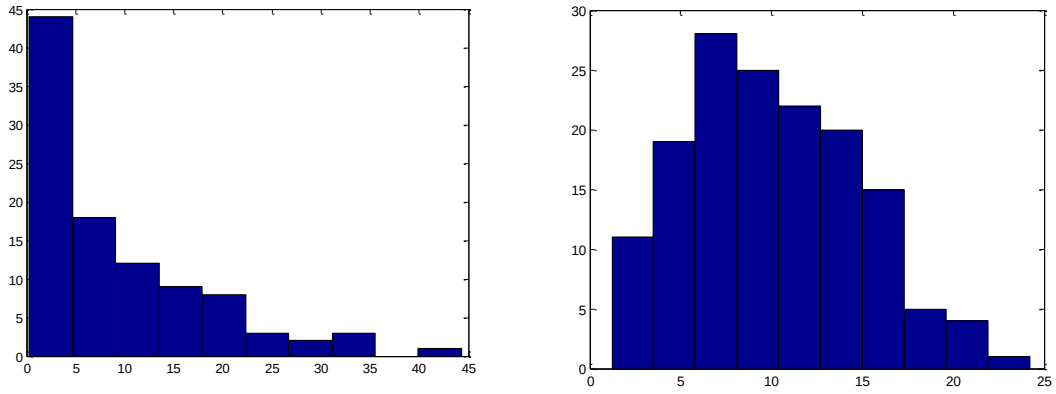



**Figure 6-1 Unimodal distributions**

Histograms that are clearly multimodal or uniform do not qualify for characterizing the system group.

### 6.1.2 Testing for uniformity

Though a visual inspection can identify the histogram has one or more modes, unimodality should be verified formally. A formal verification is needed because the shape of the distribution can be deceiving and a distribution that can be identified as unimodal can be in fact uniform. In the Figure 6-2 all histograms depict random number from a uniform distribution. All distributions of these examples can be perceived as being unimodal.

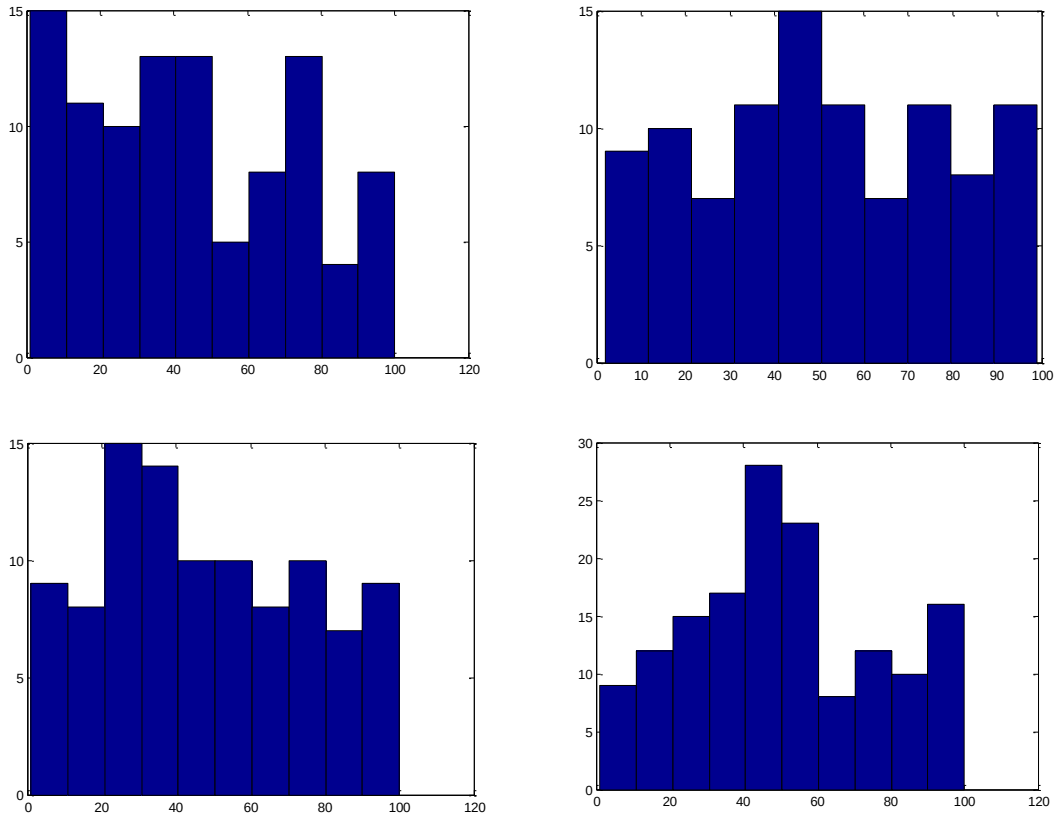



**Figure 6-2 Histograms of uniformly random numbers**

To ensure that such an error is unlikely to occur, a chi-square hypothesis test is used.

The Null hypothesis of the test is formulated as following:
$H_0$ : *"The empirical distribution is uniform"*

Whereas the alternative states:
$H_1$ : *"The empirical distribution is non-uniform".*

The test is performed with the Pearson's goodness of fit test for uniformity. The goodness of fit test helps establish whether or not an empirical distribution differs significantly from a theoretical uniform distribution. The *N* observations of the sample data are divided into $n$ cells. Each cell contains the observed frequencies $O_i$. The theoretical frequency for any cell under the null hypothesis is:

$E_i = \frac{N}{n}$, where $i = 1, 2, ..., n$.

Condition for determining the number of cells is that $E_i \geq 5$ .The test statistic is evaluated:

$$X^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \qquad \textbf{( 6-1)}$$

The Chi square statistic is used to evaluate the p-value of the test statistic $X^2$, with $n-1$ degrees of freedom, and for a significance level $\alpha$ .

### 6.1.3 Type I and II Errors and associated risks

The errors of incorrectly rejecting a true null hypothesis (Type I) or incorrectly not rejecting a false null hypothesis (Type II), come with different types of costs. These costs are assessed here in the context of this application.

Type I error leads to incorrectly applying a system group characteristic to newly installed systems. This results to erroneous processing of traces. For the segmentation process, this most likely would lead to the truncation of subsequences (generally this is a higher risk compared to collision as the latter depends greatly on the system failure rate). The truncation of subsequences would inevitably affect the association coefficient and consequently the cost function, which would mislead the tag matching process. Type I error has an impact on the consistency of the information obtained after transformation. .

Type II error on the other hand, would reject a group wide parameter value that would result to newly installed systems to be parameterized independently, which has a delay on the utilization of the transformation process.

The transformation process emphasizes the benefits gained from the utilization of traces. For effective utilization, data consistency is vital. In this work, no method is proposed that will allow the timely identification of wrongful transformation of traces of newly installed systems if a group value is applied wrongly. To be able to detect mistakes in the transformation, there has to be enough data (adequate number of subsequences) to allow the parameterization of the system following the proposed methodology. This will happen after the burn in period is over. Consequently a Type I error would lead to costs of misinformation plus the costs of Type II error. Based on that, the lower cost is identified with Type II error. To reduce the risk of Type I Error

the level of significance for the Goodness of fit test is set to $a = 0,001$. Though the significance level is set arbitrarily, it is certainly a very low probability for Type I error to occur.

In section 6.1.4 the test for uniformity is described as it applies for the cutoff parameter. In section 6.1.5 the test is described as it applies for the association coefficients.

### 6.1.4 Goodness of fit for the cut off parameter values

The segmentation process returns one value for the cutoff parameter for every sequence that has been segmented. Every sequence represents one system. For a group of $N$ systems there will be equal number of cutoff values forming the sample set. The range that these values will cover is not known in advance. To perform the hypothesis test this range has to be defined, as it will be the range where the theoretical uniform distribution will be expected.

To set the range we look into the sample dataset of the cutoff values. For a set of values $\Theta$ of the *cutoff parameter*, the range is defined by $[\theta_{\min}, \theta_{\max}]$, where $\theta_{\min} = \min(\Theta)$ and $\theta_{\max} = \max(\Theta)$. This range is divided into $n$ cells so that the condition $E_i \geq 5$ is satisfied. Given the above the test can be performed.

### 6.1.5 Goodness of fit test for association coefficients

As in the case with the cutoff parameter values, the association coefficients are tested too. However there are two main differences between the two cases.

The first difference is that there might be some pairs of semantics for which no observations are made in some systems. This is not the case for the cutoff value that is obtained given a sequence with error or recovery subsequences. For semantics to be available, the events that trigger them need to have occurred.

The second difference is that the range in which the theoretical uniform distribution is expected is known in advance i.e. $[0,1]$.

To be able to perform the test on the association coefficients from all systems in the group, the pair wise coefficients need to be arranged appropriately. To explain this arrangement, the notation of Chapter 5 is used.

Since the pool of semantics is knows in advance (the set of semantics a system can produce is a known design feature), the association matrix is formed by the orthogonal arrangement of all semantics $|IC| = w$ (here $IC$ represents the pool of semantics). This ensures that the matrices of all systems are of the same size and that the semantics are ordered in the same way (lexicographically) across all systems. For the group of $N$ systems, equal number of association matrices are arranged in tandem to form a three dimensional matrix $\Phi^C$ of size $w \times w \times N$, containing the associations $\phi_{ij}$ for the k$^{th}$ system in the two dimensional setting $w \times w$ of $\Phi_k$, where $1 \leq k \leq N$ (Figure 6-3). The associations $\phi_{ij}$ are tested across the third dimension of $\Phi^C$ (e.g. the grey cells in Figure 6-3 represent the association $\phi_{1w}$ between $ic_1$ and $ic_w$ across all $\Phi_k$

). As mentioned before not every $\phi_{ij}$ has a value depending whether some *ics* are found in the collection of tags $LC_{k_0}$ or not. Therefore the set of pair-wise associations $S\phi = \{\phi_{ij}^1, \phi_{ij}^2, \phi_{ij}^3, ..., \phi_{ij}^N\}$, where $\phi_{ij}^k \in \Phi_k$ across $N$ systems, can have a size of $0 \leq |S\phi_{ij}| \leq N$ counts.

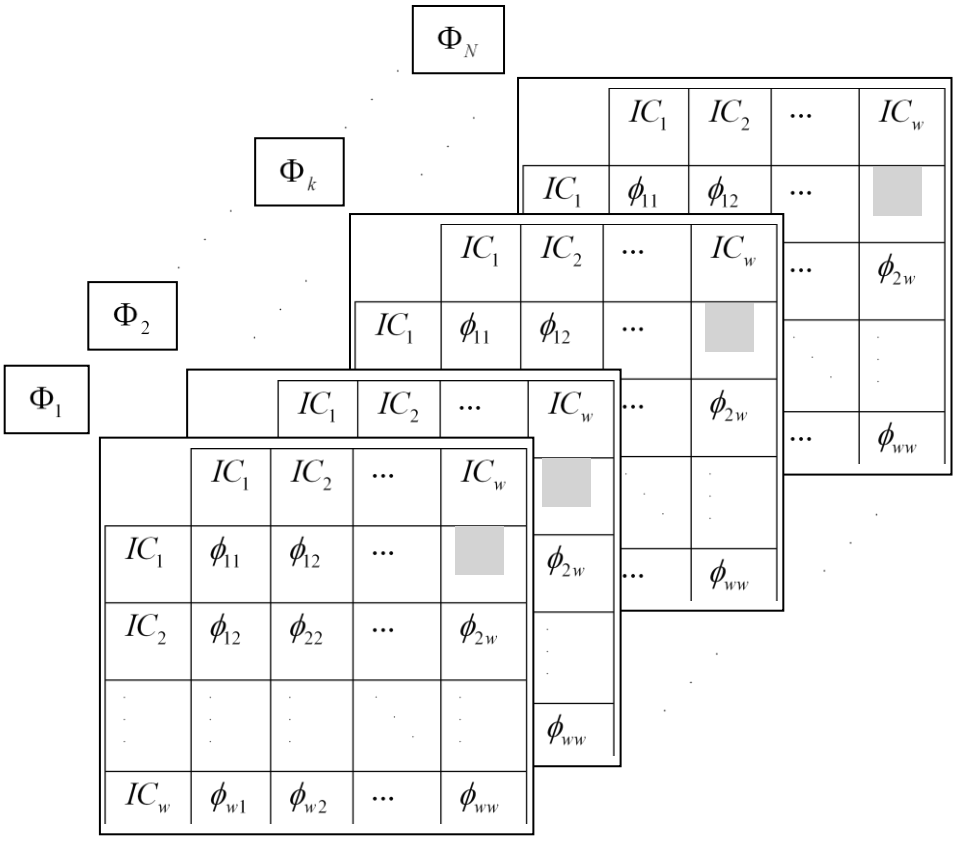



**Figure 6-3 Collective association matrix $\Phi^C$**

The association measure $\phi_{ij}$ as defined in chapter 5, is a continuous measure that takes values in $0 \leq \phi_{ij} \leq 1$. The null hypothesis is based on the discrete uniform distribution over $n$ values. To fit to that requirement, the interval $I_o = [0,1]$ is divided into $n$ non-overlapping subintervals $I_i$ of equal width $w_i$. The probability of an association $\phi_{ij}$ falling in any of the subintervals $I_i$ under the null hypothesis is $E_i = \frac{|S\phi_{ij}|}{n}$. Given the matrix $\Phi^C$ the test is performed for the pairs of semantics where the size of the set $|S\phi_{ij}|$ allows $E_i \geq 5$. The observed frequency $O_i$ is computed by counting the association coefficients $\phi_{ij}^k$ that fall within each subinterval $I_i$.

## 6.2 Discussion and conclusion

Though the search for unimodality in the histogram of the empirical distribution and the formal test that follows ensure that the parameter values show coherence the level of coherence is not tested formally.

Ideally the congruence of the parameter values would manifest itself as a compact empirical distribution, with most values clustered around a central point (low spread). For the cutoff parameter this would suggest that the distributed systems, though operating under different conditions, form subsequences with very similar temporal structure i.e. distance between traces of the same subsequence. If the point of congruence, the mode, is a low value e.g. 1-3 sec, this would suggest that the subsequences are also very compact. Similarly for the association coefficient, congruence of the coefficient values to the left or to the right of the range would be a clear sign of weak or strong association respectively.

Such observations would not only allow the characterization of group with parameter values, it would also be strong evidence of the soundness of the transformation process, because it would demonstrate that consistent results are obtained throughout independent samples of sequences.

The interpretation of the observed congruence of the parameter values in the system group is also bound to the size of the sample. The restriction put by the goodness of fit test $E_i \geq 5$, results to wider cells for smaller sample size in order to meet the criterion. Consequently, the statements about the observed congruence have to be given in respect to the width of the cells. This methodology is applied in the case study of chapter 7.

# Chapter 7

# 7 Case Study

The methodology that was presented in chapters 4, 5 and 6 is applied on a sample set of traces obtained from systems operating in the field. This chapter contains the results of the application of the proposed methodology on the sampled data set. The chapter's aim is to:

1. Demonstrate the application of the transformation process on a single sequence
2. Demonstrate the data reduction that is achieved by the transformation process on the sample set of sequences
3. Perform the test for coherence (test of uniformity) for the cutoff parameter and the associations between semantics for the sequences in the sample
4. Assess the performance of cost function in respect to the triangle looseness criterion

First an overview of the sample data set is given in 7.1. In section 7.2 the transformation methodology is applied on a single sequence. In section 7.3 the data reduction that is achieved on the sample set with the proposed methodology is presented. In section 7.4 the test of coherence is applied on the parameter values obtained from the sample set. In section 7.5 the performance of the cost function is discussed. The chapter closes with section 7.6 where the results are discussed.

## *7.1 The sample data set*

The sample sequences are taken from 137 systems. The systems are of the same type with the same software and hardware components installed. These systems are geographically distributed. The systems are used in the same area of clinical application but their actual operating conditions are not known. The sequences are cleared from all irrelevant types of traces (see preprocessing 2.5). The only types of traces that remained in the sequence are those of type error and recovery. Partially periodic subsequences are removed too using the procedure described in 3.4.

The sequences are of various lengths and contain various amounts of traces. This allows testing the transformation process under different combinations of sequence length and number of traces. In any case the segmentation, which is the most influential phase of the transformation process, should return the most compact subsequences possible given the arrangement of traces in the sequence. In Figure 7-1 the lengths of the sequences are shown in an increasing order. Their durations vary from less than 500 hours of operating time up to almost 10000 hours.

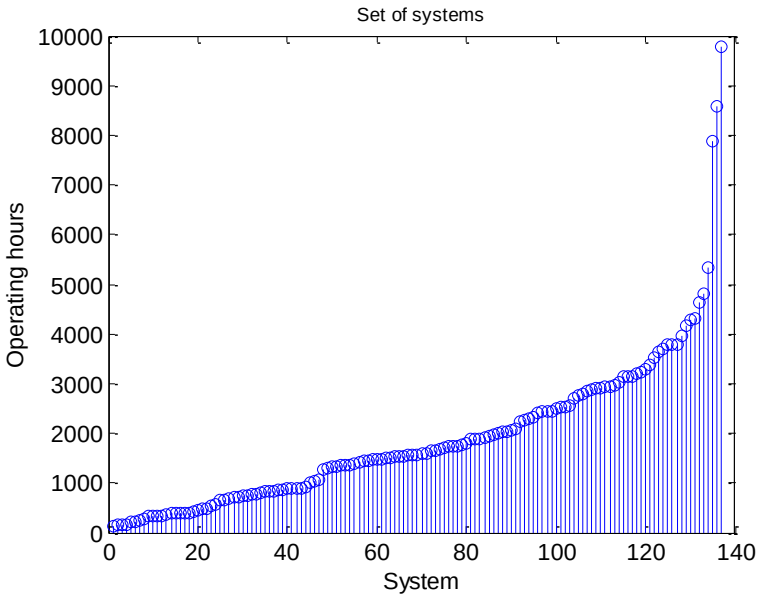



**Figure 7-1 Operating time per system**

The sequences contain a variety of number of trace. In Figure 7-2 the number of error traces is plotted against the length of the sequence. There is a weak positive correlation between the number of error traces and the length of the sequence.

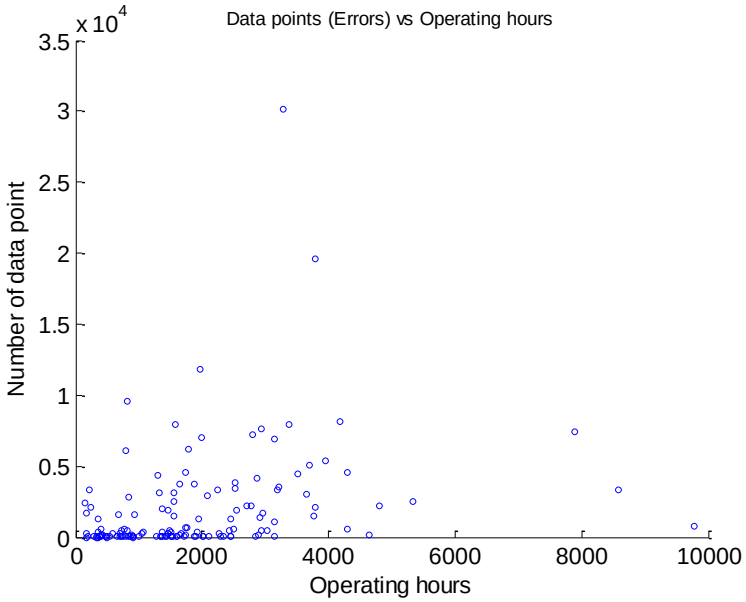



**Figure 7-2 Data points (errors) in a data sequence**

Similarly for recovery events the scatter plot of the recovery traces versus the operating time is seen in Figure 7-3. Again there is a weak correlation between the number of recovery traces logged and the length of the sequence.

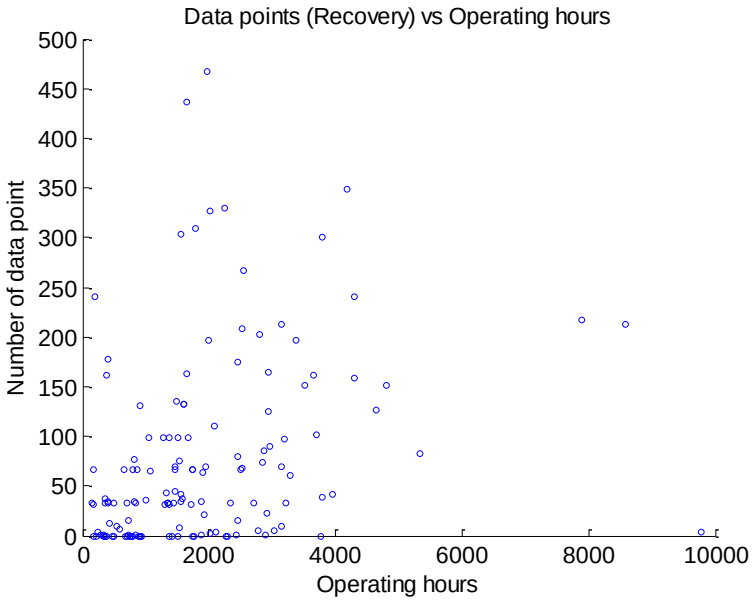



**Figure 7-3 Data points (recovery) in a data sequence**

The absence of strong correlation between the number of traces found in a sequence and the length of the sequence is an indication of how the location and system specific factors, such as the type of events, the workload, the operating conditions, system configuration etc. can contribute to the amount of traces logged. This is a reminder of how subsequences can vary from one system to another in the number of traces and type of semantics they can contain. The observation made here is in line with the observations made in the fault injection experiment (3.2).

## *7.2 Transformation methodology applied on a single sequence*

In this example the untransformed sequence has a length of over 2200 hours of operating time. The example follows the framework of 2.4.3. Selection has taken place with choosing the sample set. The steps that follow are preprocessing, segmentation, tagging and tag matching.

### 7.2.1 Preprocessing

First all traces that are of type other than "Error" are removed. This is done automatically using a script that filters out unwanted events.

After removal of all unwanted events, the sequence contains 21970 error traces. A graphical representation of the sequence is shown in Figure 7-4.

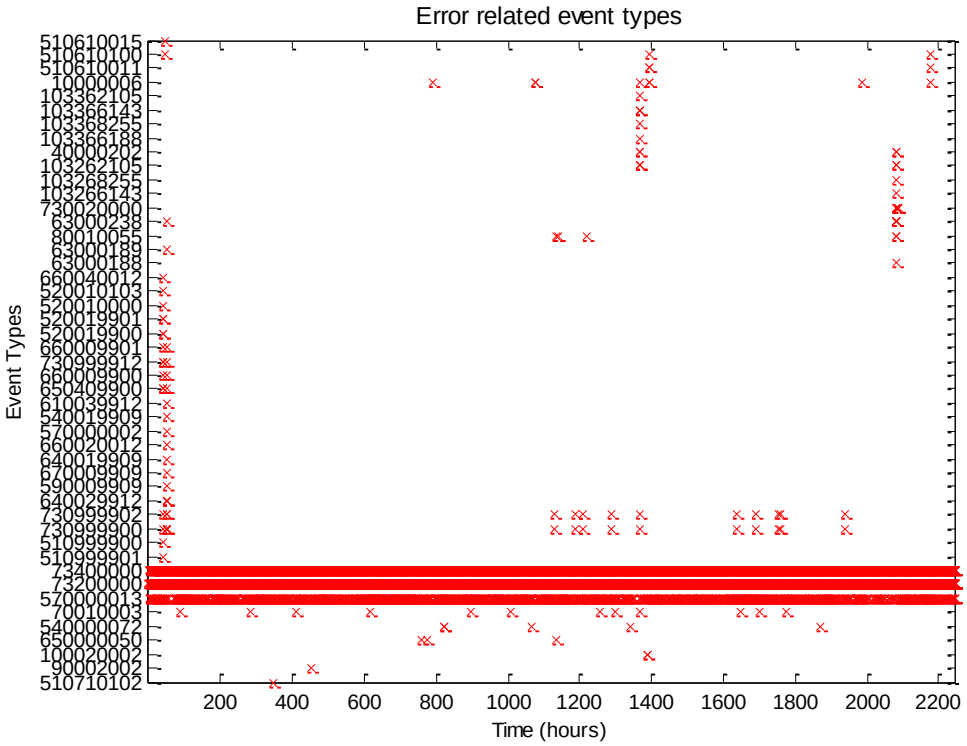



**Figure 7-4 Graphical representation of error sequence before pps are removed**

As part of the preprocessing step, all *ics* of error traces are tested if they are *pps*. The result of the tests shows that three *ics* are *pps*, namely 73400000, 7320000 and 570000013. The pps are removed from the sequence. After the removal of *pps* the sequence contains 489 error traces. The preprocessed sequence is show in Figure 7-5

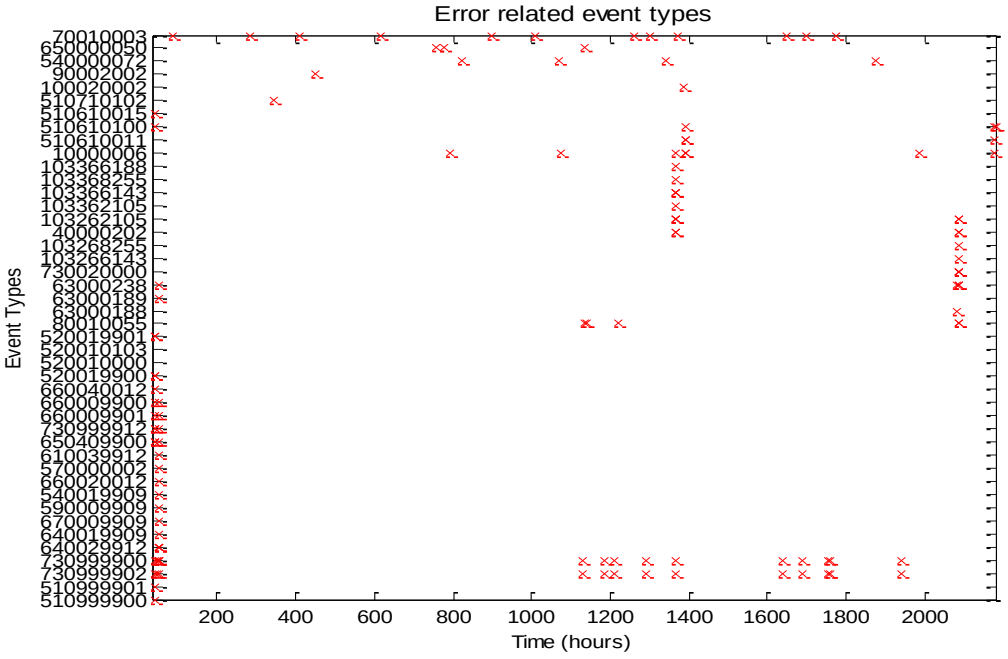



**Figure 7-5 Error sequence after preprocessing is completed**

The sequence is shown in Figure 7-5 using the visualization method described in 3.3.

### 7.2.2 Segmentation step

The segmentation process consists of five elements:

1. Use a sequential clustering algorithm to segment the sequence and obtain multiple segmentation results for a range of values for the cutoff criterion. (see method in 4.2.1)
2. Find which values of the cutoff criterion give the best segmentation result using a cluster separation measure (see method in 4.2.2).
3. Robustify the value of the cutoff criterion to account for structural variation using the resampling method and the *hCSM* (see method in 4.2.3).
4. Chose the adjusted value of the cutoff criterion that reduces the risk of collision for application (see method in 4.2.4).
5. Perform internal validation to verify that the segmentation of the sampled sequence using the selected cutoff value is not the result of data points being grouped randomly (see method in 4.2.5).

#### 7.2.2.1 Segmentation (elements 1 and 2)

The sequence is segmented using the sequential clustering algorithm. The range of values for the cutoff parameter is Θ=[0,1000]. A CSM value is obtained for the cluster result of each value of Θ (theta). The plot of the CSM values over the range of Θ is shown in Figure 7-6.

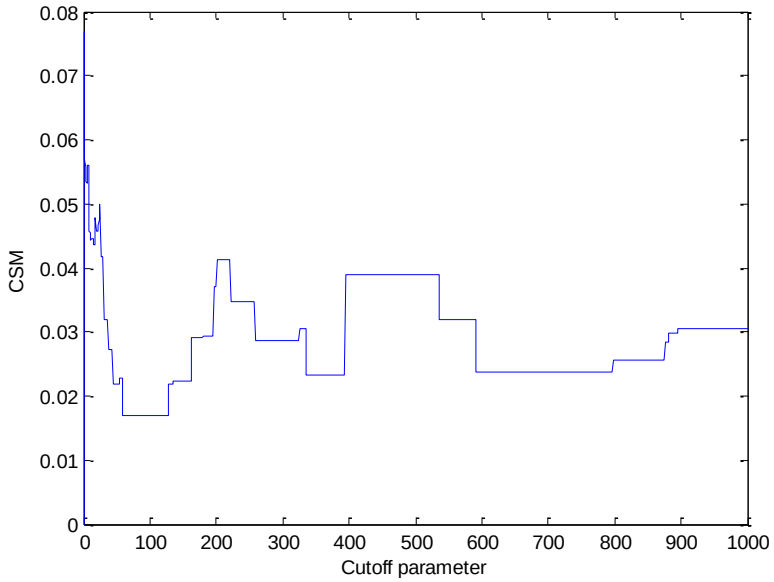



**Figure 7-6 Values of CSM over the range of theta**

The lowest CSM value(s) indicate which segmentation result returns the most compact clusters. The lowest value of the CSM is found in the range cutoff_low_CSM = [60, 128], where CSM=0.0169. Within this range the number of the identified subsequences is 68. At this point the compression ratio achieved with the segmentation is $cs_r = 0.1391$

#### 7.2.2.2 Robustification (element 3)

The resampling method is applied on the range of best segmentation results as indicated by the cutoff parameter, cutoff_low_CSM = [60, 128]. The mean distance between successive traces of the subsequences is mDBST=2.8519, therefore the

preferred thinning probability is$f = 0.3$ (see 4.2.3.2). The sequence is resampled 1000 times and the average figure of merit obtained $M_L$ from the procedure. The results of the resampling method can bees in Figure 7-7 (blue curve).

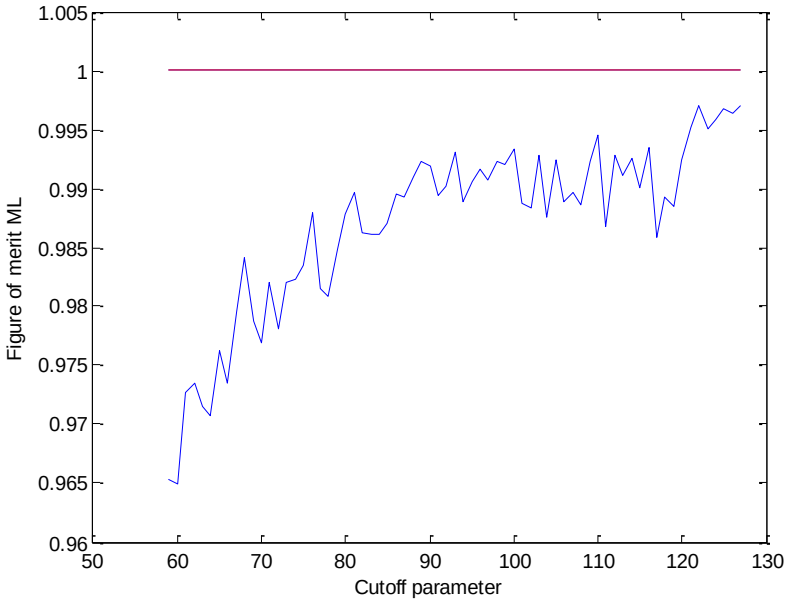



**Figure 7-7 Figure of merit ML over the best range of cutoff parameter values**

For the entire range of the cutoff parameter the figure of merit $M_L$is high (>0.965), which suggests robustness to variation. However, in the low end of the range the clustering of the resampled sequences resulted to more truncations than in the high end. The risk of truncation is lowest for the values of the cutoff parameter that are at the high end of the range.

Using the CSM and the figure of merit the hCSM is calculated. The hCSM is shown in Figure 7-8.

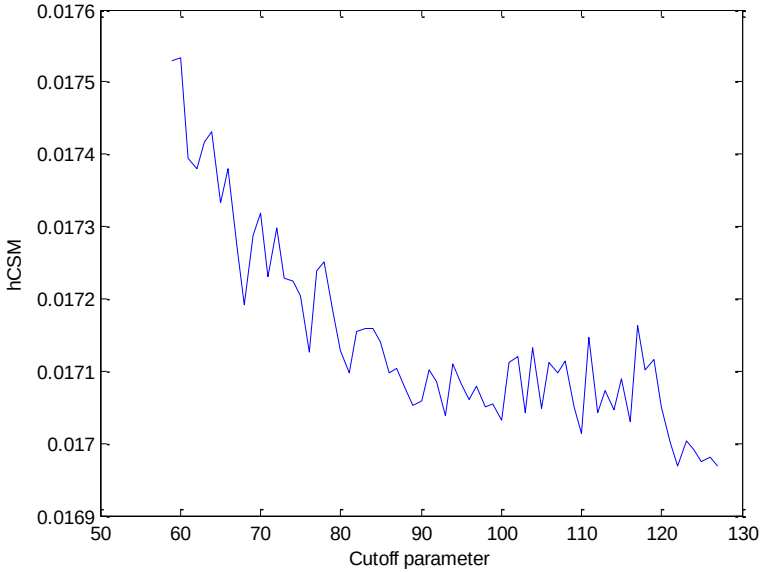



**Figure 7-8 hCSM values over the best range of the cutoff parameter values**

The lowest values of the hCSM are found for two values of the cutoff parameter, $\theta_1 = 124$ and $\theta_2 = 128$.

### 7.2.2.3 Collision probability (element 4)

To estimate the collision probability when the algorithm is applied in the field, the empirical distribution of the length of the subsequences is needed. The lengths of subsequences are taken directly from the segmentation result. The empirical distribution of the lengths of subsequences for the sequence in the example is shown in Figure 7-9.

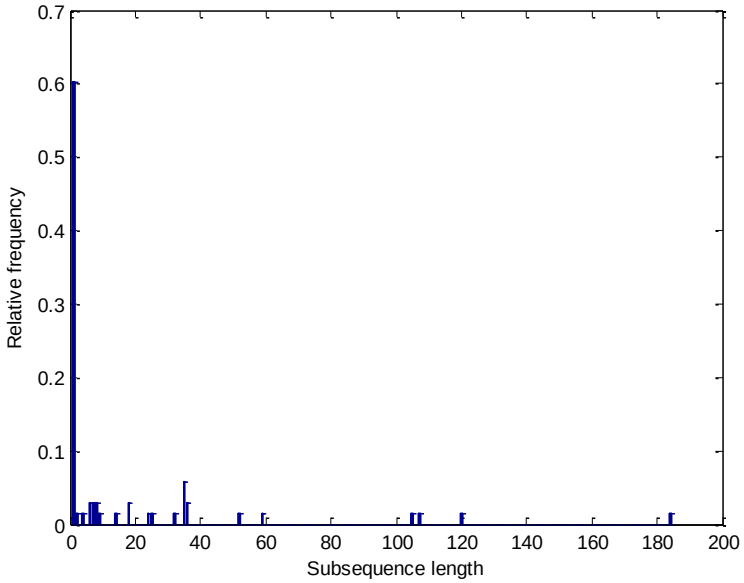



**Figure 7-9 Empirical distribution of lengths of subsequences**

Using the method described in 4.2.4, the collision probability is estimated for the chosen range of values of the cutoff parameter. The collision probability for this range is shown in Figure 7-10 Collision probability for selected range of cutoff parameter values. The collision probability is overall very low, but as expected it is increasing as the value of the cutoff parameter is increasing. The effect of the collision probability is naturally working in favor of the lowest value $\theta_1 = 124$ of the cutoff parameter. The value $\theta_1 = 124$ is the one that should be used when the sequence of traces for this system is segmented in field application.

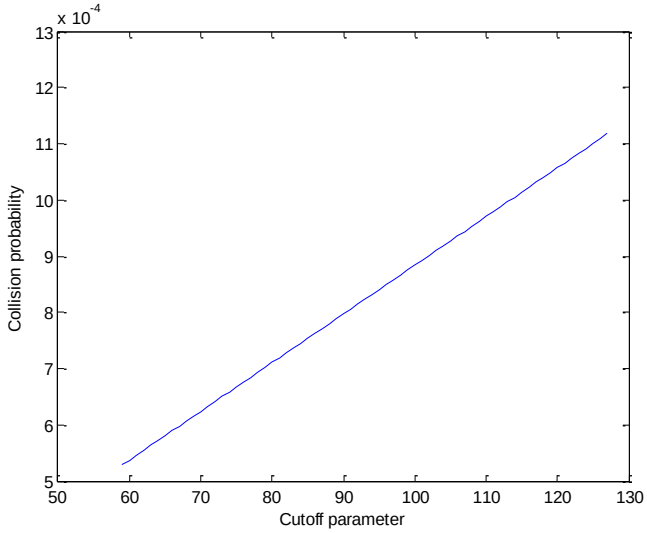



**Figure 7-10 Collision probability for selected range of cutoff parameter values**

#### 7.2.2.4 Random position hypothesis test (element 5)

The test is performed by first calculating the $\hat{\Gamma}$ statistic. For the sequence in the example the statistic is $\Gamma = 0.1948$. The empirical distribution of $\hat{\Gamma}_r$ is computed from 5000 random positioning sequences, where 486 data points are randomly generated in an interval equal to the length of the original sequence in the sample. The empirical distribution can be seen in Figure 7-11 (blue histogram). The statistic $\hat{\Gamma}$ for the sequence in the example is larger than all $\hat{\Gamma}_r$ (red line indicates the value of $\hat{\Gamma}$), therefore the null hypothesis can be rejected. It is very unlikely that the clustering result obtained by the segmentation of the original sequence can be obtained by a sequence with randomly positioned traces. The clustering result is a sensible result based on the non-random arrangement of traces in the sequence.

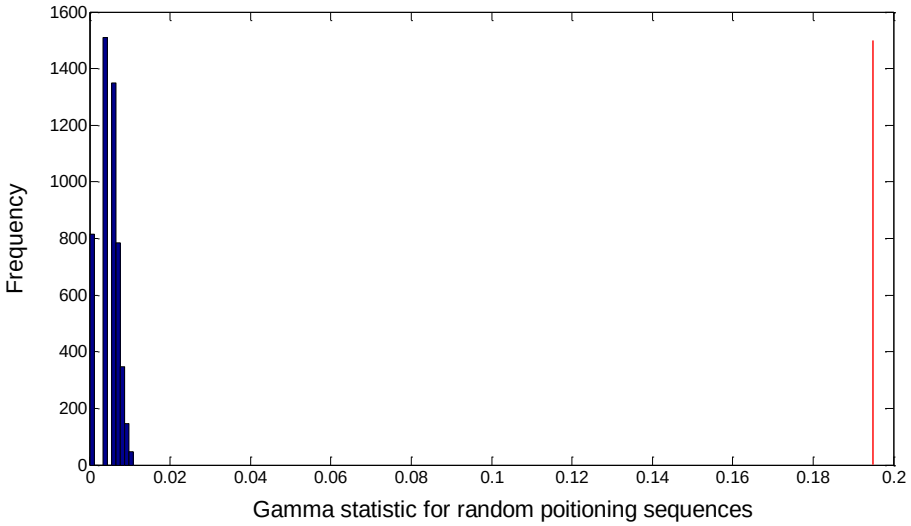



**Figure 7-11 Empirical distribution of gamma statistic under the null hypothesis**

### 7.2.3 Tagging

The tagging operation eliminates the replicates of semantics in the subsequences, it is ordering the semantics lexicographically and it is assigning point representations to the subsequences. An example of a tagging of a subsequence is shown below. The subsequence after the segmentation (A) contains 12 traces of 5 different types (*ics*):

**A. Subsequence after the segmentation**

| | | | | | | |
|---|---|---|---|---|---|---|
| 10000006 | 10000006 | 40000202 | 103262105 | 103366143 | 103366188 | 103366143 |

| | | | | |
|---|---|---|---|---|
| 103368255 | 103366143 | 103366143 | 40000202 | 103262105 |

**B. After the tagging of the semantics (eliminate replicates and ordering):**

| | | | | |
|---|---|---|---|---|
| 103262105 | 103362105 | 103366143 | 103366188 | 103368255 |

After the tagging of the semantics the subsequence contains only 5 types of traces, which are ordered. The temporal information is tagged by assigning to eat subsequence the time of occurrence of the first trace in the subsequence.

With the tagging of the semantic and temporal information, the sequence contains only tags. The result of the tagging of subsequences can be seen in Table 7-1. The

table contains all tag in the sequence. The tags are presented in order of occurrence. The time of occurrence is given in the third column of the table labeled as "Times". The fourth column shows the time between tags (TBT). The TBT shows that most tags are separated from each other by relatively long intervals.

| Index | Tags | Times (sec) | TBT (sec) |
|---|---|---|---|
| 1 | <520010000, 520010103, 520019900, 520019901, 650409900, 660009900, 660009901, 660040012, 730999912> | 0 | |
| 2 | <510999900, 510999901, 650409900, 730999900, 730999902> | 1579 | 1579 |
| 3 | <510610015, 510610100> | 17975 | 16396 |
| 4 | <730999900, 730999902> | 30895 | 12920 |
| 5 | <540019909, 570000002, 590009909, 610039912, 640019909, 640029912, 650409900, 660009900, 660009901, 660020012, 670009909, 730999900, 730999902, 730999912> | 37544 | 6649 |
| 6 | <640029912> | 38603 | 1059 |
| 7 | <63000189, 63000238> | 42990 | 4387 |
| 8 | <70010003> | 167658 | 124668 |
| 9 | <70010003> | 871503 | 703845 |
| 10 | <510710102> | 1098922 | 227419 |
| 11 | <70010003> | 1337862 | 238940 |
| 12 | <90002002> | 1478041 | 140179 |
| 13 | <70010003> | 2073731 | 595690 |
| 14 | <650000050> | 2581353 | 507622 |
| 15 | <650000050> | 2653180 | 71827 |
| 16 | <10000006> | 2706767 | 53587 |
| 17 | <540000072> | 2812450 | 105683 |
| 18 | <540000072> | 2812646 | 196 |
| 19 | <540000072> | 2812982 | 336 |
| 20 | <70010003> | 3083172 | 270190 |
| 21 | <70010003> | 3491866 | 408694 |
| 22 | <70010003> | 3492001 | 135 |
| 23 | <540000072> | 3702845 | 210844 |
| 24 | <10000006> | 3729326 | 26481 |
| 25 | <10000006> | 3730245 | 919 |
| 26 | <730999900, 730999902> | 3924446 | 194201 |
| 27 | <80010055> | 3947995 | 23549 |
| 28 | <650000050> | 3948141 | 146 |
| 29 | <80010055> | 3955251 | 7110 |
| 30 | <730999900, 730999902> | 4131265 | 176014 |
| 31 | <730999900, 730999902> | 4206195 | 74930 |
| 32 | <80010055> | 4248512 | 42317 |
| 33 | <70010003> | 4386967 | 138455 |
| 34 | <730999900, 730999902> | 4500504 | 113537 |
| 35 | <70010003> | 4540707 | 40203 |
| 36 | <540000072> | 4692482 | 151775 |
| 37 | <10000006, 40000202, 103262105, 103362105, 103366143, 103366188, 103368255> | 4777343 | 84861 |
| 38 | <730999900, 730999902> | 4781481 | 4138 |
| 39 | <70010003> | 4785630 | 4149 |
| 40 | <100020002> | 4847227 | 61597 |
| 41 | <100020002> | 4848112 | 885 |
| 42 | <10000006, 510610011, 510610100> | 4871819 | 23707 |
| 43 | <730999900, 730999902> | 5746691 | 874872 |
| 44 | <70010003> | 5785741 | 39050 |
| 45 | <730999900, 730999902> | 5934973 | 149232 |
| 46 | <70010003> | 5975885 | 40912 |
| 47 | <730999900, 730999902> | 6167294 | 191409 |
| 48 | <730999900, 730999902> | 6186146 | 18852 |
| 49 | <70010003> | 6240492 | 54346 |
| 50 | <540000072> | 6596103 | 355611 |
| 51 | <730999900, 730999902> | 6841564 | 245461 |

| | | | |
|---|---|---|---|
| 52 | <10000006> | 6998704 | 157140 |
| 53 | <63000188, 63000238> | 7348973 | 350269 |
| 54 | <63000238> | 7349132 | 159 |
| 55 | <63000238> | 7349360 | 228 |
| 56 | <63000238> | 7356978 | 7618 |
| 57 | <730020000> | 7357244 | 266 |
| 58 | <730020000> | 7357442 | 198 |
| 59 | <80010055> | 7358275 | 833 |
| 60 | <40000202, 63000238, 103262105, 103266143, 103268255, 730020000> | 7358811 | 536 |
| 61 | <63000238> | 7359325 | 514 |
| 62 | <80010055> | 7359531 | 206 |
| 63 | <730020000> | 7359856 | 325 |
| 64 | <730020000> | 7360481 | 625 |
| 65 | <730020000> | 7362617 | 2136 |
| 66 | <730020000> | 7363918 | 1301 |
| 67 | <730020000> | 7364133 | 215 |
| 68 | <10000006, 510610011, 510610100> | 7700983 | 336850 |

**Table 7-1 Tags in the sequence**

### 7.2.4 Tag matching

The tag matching operates on the tags seen in Table 7-1. The clustering algorithm for the matching operation is terminated for a silhouette value of 0.9716 (Figure 7-12) where a satisfactory clustering result was obtained. The clustering result is satisfactory because the number of tag types that are produced is low (high data reduction) and the clustering cutoff criterion value is high. The tag types are shown in Table 7-2 (horizontal axis represents the number of tag types, vertical axis represents the silhouette value).

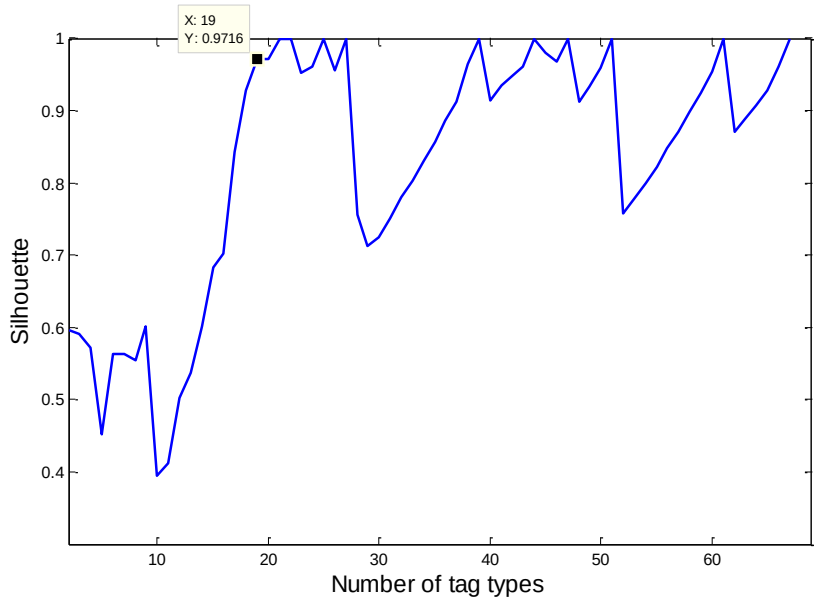



**Figure 7-12 Silhouette clustering stopping criterion**

Given the distances between the tags the tag matching operation returns 19 tag types. These are shown in Table 7-2, achieving another compression of $cr_M = 0.2794$.

| Tag Type | Tag type members |
|---|---|
| **1** | <510999900, 510999901, 650409900, 730999900, 730999902> |
| **2** | <520010000, 520010103, 520019900, 520019901, 650409900, 660009900, 660009901, 660040012, 730999912> |
| **3** | <730999900, 730999902> |
| **4** | <540019909, 570000002, 590009909, 610039912, 640019909, 640029912, 650409900, 660009900, 660009901, 660020012, 670009909, 730999900, 730999902, 730999912> |
| **5** | <640029912> |
| **6** | <80010055> |
| **7** | <63000188, 63000238> |
| | <63000189, 63000238> |
| | <63000238> |
| **8** | <730020000> |
| **9** | <40000202, 63000238, 103262105, 103266143, 103268255, 730020000> |
| **10** | <10000006, 40000202, 103262105, 103362105, 103366143, 103366188, 103368255> |
| **11** | <10000006, 510610011, 510610100> |
| **12** | <10000006> |
| **13** | <510610015, 510610100> |
| **14** | <510710102> |
| **15** | <100020002> |
| **16** | <90002002> |
| **17** | <540000072> |
| **18** | <650000050> |
| **19** | <70010003> |

**Table 7-2 Tags grouped into tag types**

In the clustering result of this example the tag types that are produced can be categorized in three cases. First is the case were the content of the tag type is a single tag that has occurred only once. Such tag types are 1, 2, 4, 9 and 10. These tags contain few common traces but mostly contain traces that are unique for that tag. There is not enough information on similarities for these traces to help cluster the tags together. This case does not contribute to data reduction as a single tag instance defines the tag type. The second case is tag types that contain a single tag that has occurred multiple times. Such tag types are 3, 5, 6, 8, 11, 12, 13, 14, 15, 16, 17, 18, and 19. The tag types resulted from the fact that their *edit distance* is equal to zero as they are exact matches. In this example these tag types are mostly responsible for the data reduction as they have taken multiple instances of identical tags and grouped them. The third case is the tag types where different tags are grouped in the same tag type because of their similarity based on the *edit distance*. The tag type 7 is falls case. Tag type 7 contains three different tags. Though the common occurrence of the trace 63000188 with 63000238 as well as the common occurrence of 63000189 with 63000238 results to a similarity between the traces of the pairs that is greater than zero, in this case the strongest factor for the clustering of the tags into the same tag type is that these tags are short and share the same trace namely 63000238. The edit transformation from the tag <63000188, 63000238> to the tag <63000238> for example is achieved most efficiently by deleting 63000189and replacing 63000238 with itself.

At this point the transformation process is complete. The sequence is represented by only 19 tag types and 68 points in time as sown in Figure 7-14. The visual representation of the sequence in Figure 7-5 can be compared against the representation of the sequence of tags. The correspondence between Figure 7-13 and Figure 7-14 is made with the help of Table 7-2. The vertical formations of traces in Figure 7-13 are represented by the instances of tag types Table 7-2.

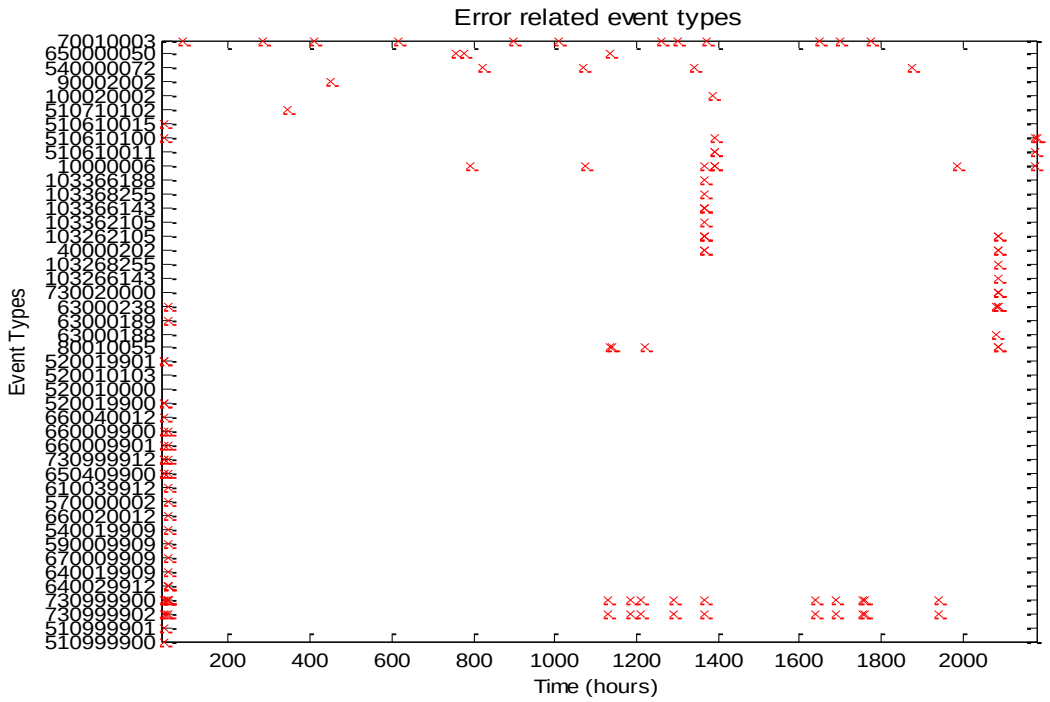



**Figure 7-13 Error sequence after preprocessing**

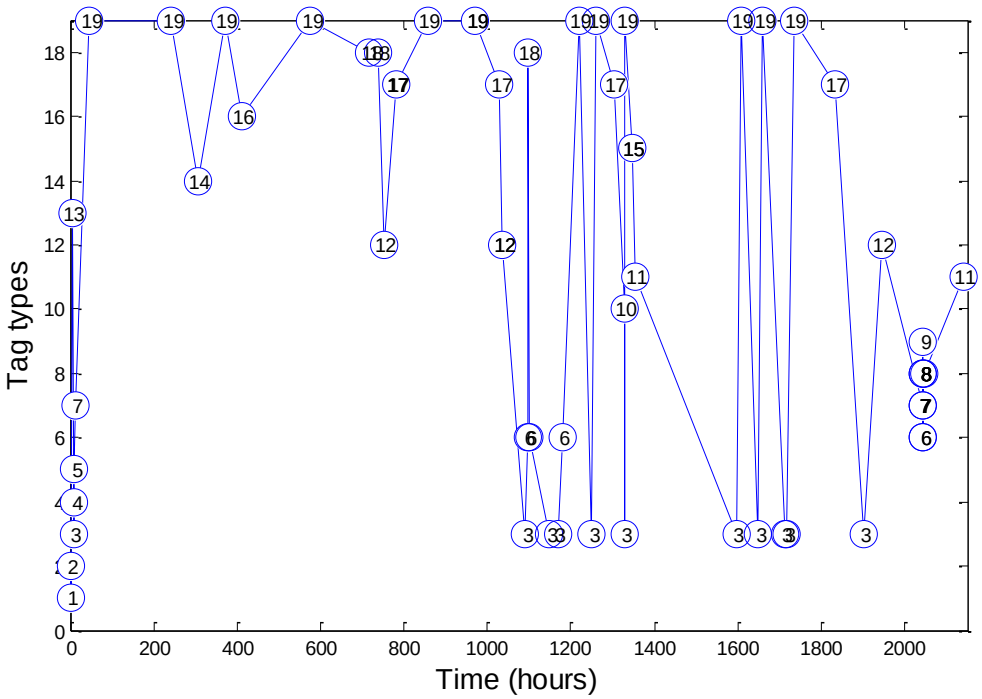



**Figure 7-14 Transformation complete: sequence of tag types**

## 7.3 Data reduction on sample sequence

Since the main objective of this methodology is to reduce the size of data, the data compression that is achieved by applying the methodology to the entire sample set is presented. The results are organized according to the type of traces.

### 7.3.1 Error Traces

Out of the 137 sequences, 9 did not contain any traces after preprocessing. From the 128 sequences that were left, three sequences returned singleton clusters i.e. subsequences containing a single trace. These data sequences are not included in the analysis as they pose no interest for the transformation process.

**1. Compression achieved by segmentation**

In Figure 7-15 it can be seen that the majority of the sequences are compressed to a rate lower than 0.05, which equals to a reduction of a sequence that originally would contain 100 traces to a sequence that contains 5 subsequences.

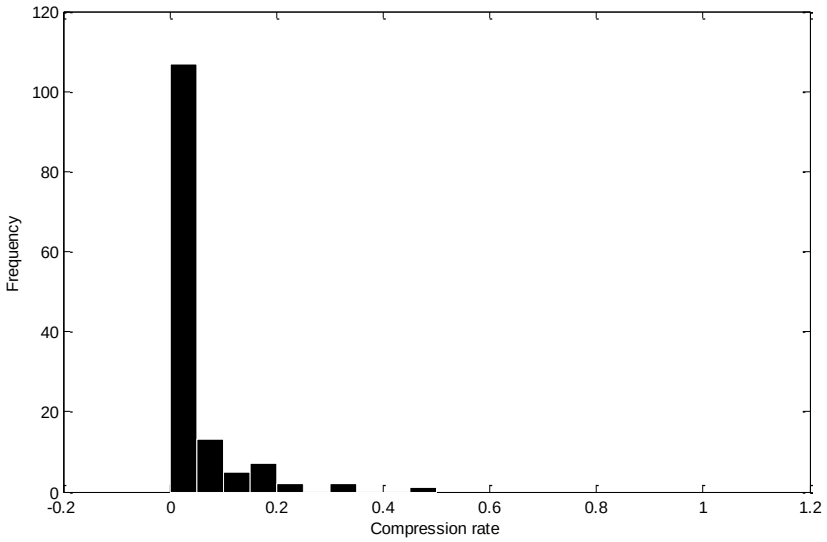



**Figure 7-15 Compression ratios after segmentation**

**2. Compression achieved with tag matching**

The tag sets obtained from the sequences after the segmentation are examined for matching tags. Tag sets that contain less than 20 observations are not considered because for none of the associations of the semantics there can be any statistical significance. This reduces the number of tag sets from 128 to 53. The matching process is performed as described in Section 5.2.2. The matching process is applied on each of the 53 tag sets. The clustering operation is stopped for high values of the silhouettes stopping rule (at least >0.9). The distribution of the compression ratio after tag matching for sample set is shown in Figure 7-16.

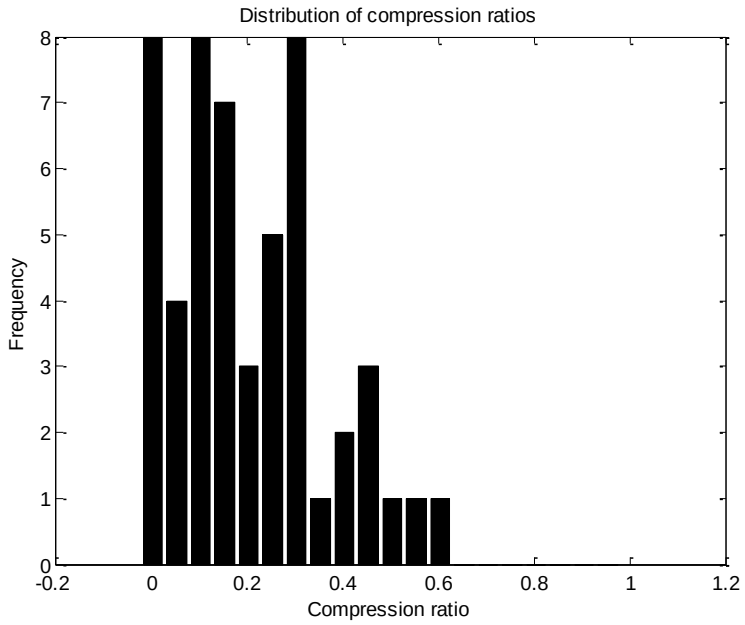



**Figure 7-16 Distribution of compression ratios after tag matching**

For the majority of tag sets a compression of ratio of $< 0,2$ was achieved. That amounts to a considerable compression in the tag set. For example a sequence that initially consisted of 100 tags can now be described by only 20 tag types, without any loss of semantic of temporal information.

### 7.3.2 Recovery traces

The procedure is repeated for the sample sequences, but this time the sequences contain recovery traces only (error traces are filtered out). Out the 137 sequences, 24 were found not to contain any traces once the partially periodic events were removed. For 16 sequences singleton clusters were returned. These sequences are not examined further.

**1. Compression achieved by segmentation**

The compression ratios achieved after segmentation is shown Figure 7-17.

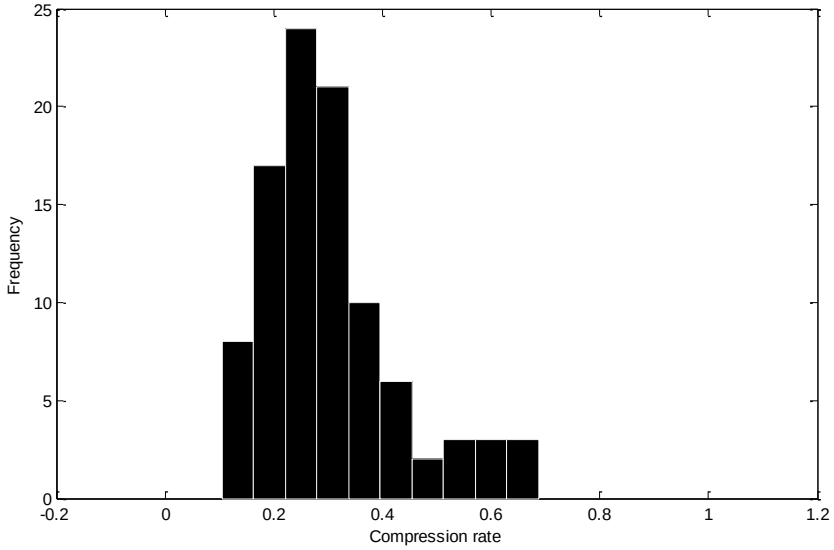



**Figure 7-17 Compression ratios after segmentation**

## 2. Compression achieved with tag matching

The matching process is applied on the tags of recovery traces. The Silhouette stopping criterion is used to recover the resulting class labels. Given the requirement of at least 20 subsequences in the set, only 5 systems are examined. For the rest of the sequences, the observed subsequences were fewer than 20. The compression ratio for the recovery related label set can be seen in Figure 7-18.

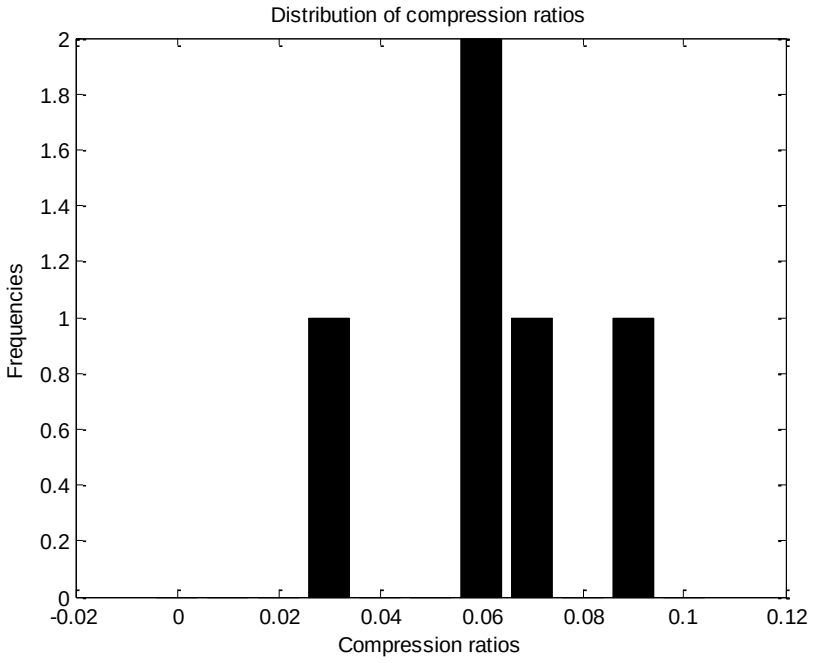



**Figure 7-18 Distribution of compression ratios for recovery related label sets**

The inequality looseness test produced also positive result for the performance of the distance function. Only for 3 out of the 8 labels sets there were a percentage of failed triplets. Within this set of 3 the highest percentage of failed triplets being 0,0203%, which is extremely low. The violation of the triangular inequality criterion is minimal.

## *7.4 Test for uniformity*

All sequences of the sample set have been segmented and the cost function for each tag set has been computed. At this point the values of all parameters for testing the coherence of the characteristics of the subsequences across the systems are available. The tests are performed separately for sequences of error and recovery traces.

### 7.4.1 Test for temporal characteristic of the group

For the temporal characteristic of systems in the sample the test for uniformity is performed using the empirical distribution of the values of the cutoff parameter across systems that were obtained with the segmentation. The test is performed using the methodology in 6.1.4

**1. Error traces**

The histogram of the cutoff values the sequences of error traces is seen in Figure 7-19. By visual inspection the empirical distribution is clearly unimodal. The set of 126 values $\theta_u$ spans over a range from $\min(\theta_u) = 1$ to $\max(\theta_u) = 784$.

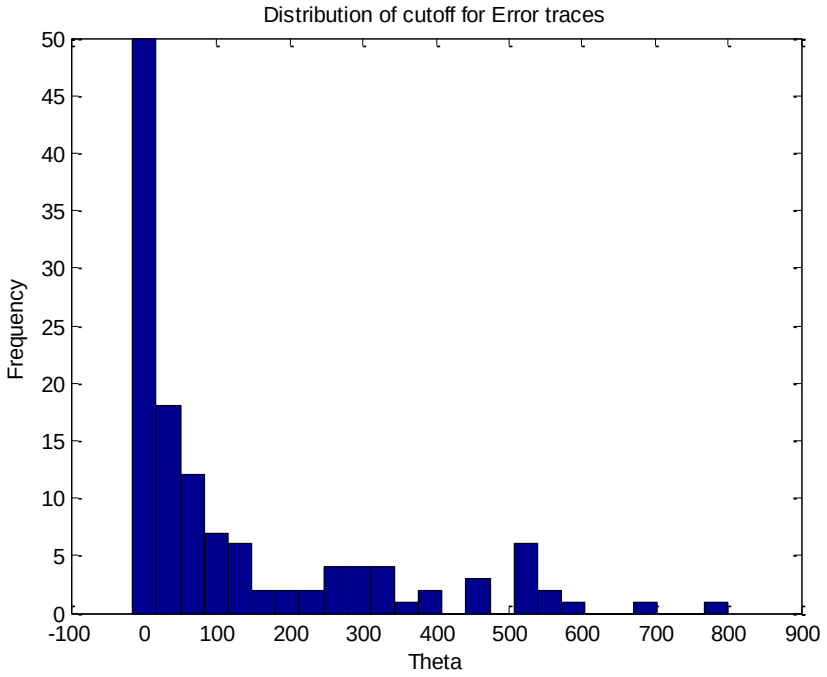



**Figure 7-19 Histogram of $\theta_u$ for 128 systems**

To meet the criterion of more than 5 expected observations per cell, the set of cutoff values is grouped in 25 non overlapping cells with width 32 seconds. Given this grouping of the cutoff values, the test statistic is calculated to be $X^2 = 627{,}6$. The null hypothesis of uniformity is rejected with a p-value of $7.4551 \text{x} 10^{-118}$.

**2. Recovery traces**

The histogram of the cutoff values the sequences of error traces is seen in Figure 7-20. By visual inspection the histogram is unimodal. The set of 95 values $\theta_u$ spans over a range from $\min(\theta_u) = 3$ to $\max(\theta_u) = 780$

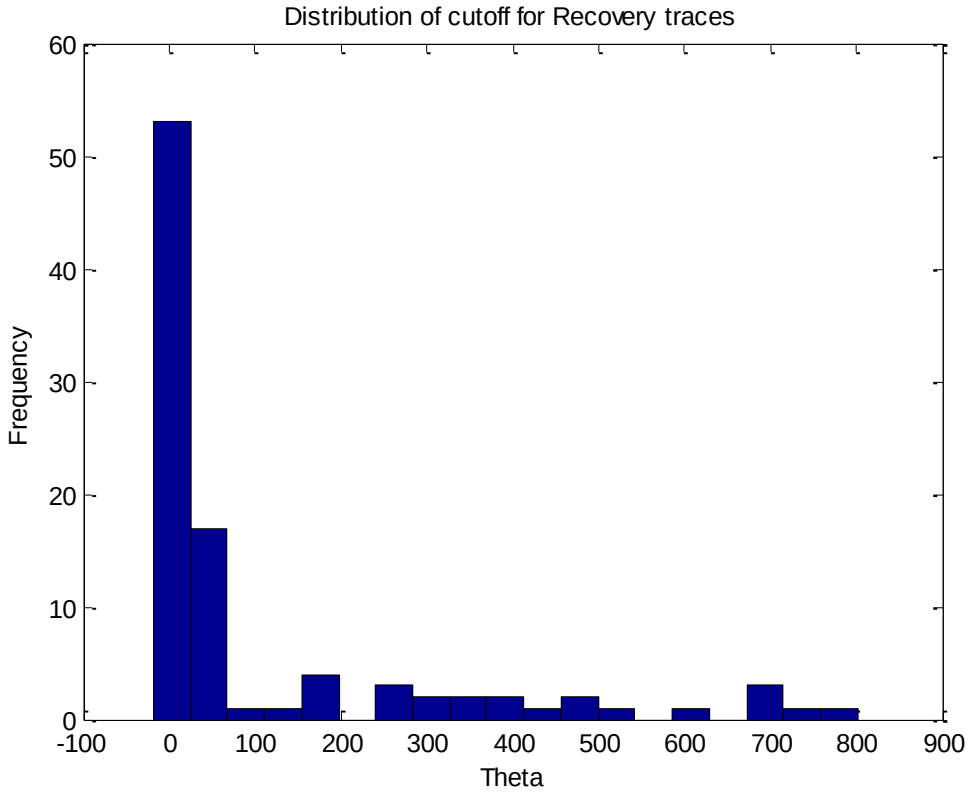


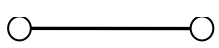

**Figure 7-20 Histogram of $\theta_u$ for 95 systems**

To meet the criterion of more than 5 expected observations per cell, the set of cutoff values was grouped in 19 non overlapping cells of width 43,667 seconds. Given this grouping of the cutoff values, the test statistic for this set is $X^2$ =1106,3. The null hypothesis of uniformity is rejected with a p-value of $1.5088 \times 10^{-224}$.

### 7.4.2 Test for the association of semantics across the group

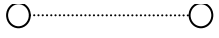
As in the case with the cutoff parameters, here the coefficients are handled separately for error and recovery semantics. To make the results easier to present an additional step is introduced. If the uniformity test is rejected the association between two semantics is classified as either *weak* (the distribution is unimodal and positive skewed) or *strong* (the distribution is unimodal and negative skewed). A strong association is represented by a solid line connecting the two *ics* (ics are represented by single digits to mask the real ic numbers).

1 2



A weak association is represented by dotted line connecting the two *ics*.

1 2



**1. Error traces**

For the test, labels sets that contain 20 or more labels are chosen. This requirement is set for the significance of the association coefficient. This requirement reduces the

number of tag sets from 137 to 53. For the remaining labels sets, the associations are inserted into the global association matrix $\Phi_k^F$ as described in 6.1.5.

To meet the requirement that the expected number of observations in each cell is greater than 5, the range 0 to 1 is divided into 10 cells. At significance level $a = 0.05$ and $df = m - 1 = 9$, the chi square distribution returns the value $\chi^2_{0.95,9} = 16.9190$. From the 53 tag sets, 524 association coefficients are computed. Not all of them are found in every system. Out of the 524 coefficients, only 68 are large enough (found in more than 50 systems) to perform the test.

For 10 out of the 68 pairs the null hypothesis for uniformity can be rejected. The associations of ICs that are consistent across systems are shown in Figure 7-21.

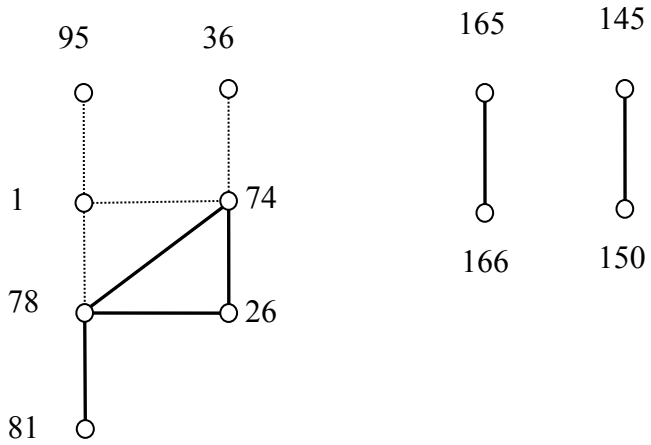



**Figure 7-21 Consistent associations of error ICs across distributed systems**

In the graph of Figure 7-21 it can be seen how the semantics are associated with each other in a consistent manner across the systems of the sample. There are four associations (1, 78), (1, 74), (1, 95), (36, 74) that are consistently *weak*. This suggests that these semantics are never or rarely found in the same tag. There are six *strong* associations found in the results. Two of them, (145, 150) and (165,166), appear as 2-tuples. There is one 3-tuple (74, 78, 26) that is consistently *strong* across all systems. The associations between semantics 26, 78 and 81 are found to be strong for (26, 78) and (78, 81) but no consistent result is found the pair (26, 81). The strong associations between error semantics are verified by using the "Software Architecture Specification" document [Phi06] of the system. The semantics with coherent strong association across systems, trace back to components that belong to the same module.

## 2. Recovery Traces

Out of the 137 tag sets, only 9 consist of more than 20 tags. These 9 tag sets form a set of 94 associations $S\phi_{ij}$. Out of these none contains more than 50 observations. The uniformity test could not be performed because of the small sample size (this applies for the grouping of data into 10 cells).

## 7.5 Performance of cost function

For the clustering of tags into tag types the performance of the normalized edit function $nD(L_i, L_j)$ in respect to the triangle inequality criterion is interesting because inform whether the comparison is reliable. The performance of the edit function is measured by the triangle inequality looseness. The triangle inequality looseness is computed for each triplet of labels $(L_i, L_j, L_k)$ in label set $LC_0$:

$$F(L_i, L_j, L_k) = nD(L_i, L_j) + nD(L_j, L_k) - nD(L_i, L_k)$$

The performance of the edit function is defined by the percentage of triples that do not satisfy the inequality looseness among all triples in a label set.

During the tag matching of the tag sets, all triplets were tested for the inequality looseness. The number of triplets that failed to meet the triangle inequality over the entire number of triplets tested can be seen in the Figure 7-22. The triples that pass the test are on the right hand side (black bars) of the zero point and the triplets that failed the test on the left hand side (red bar).

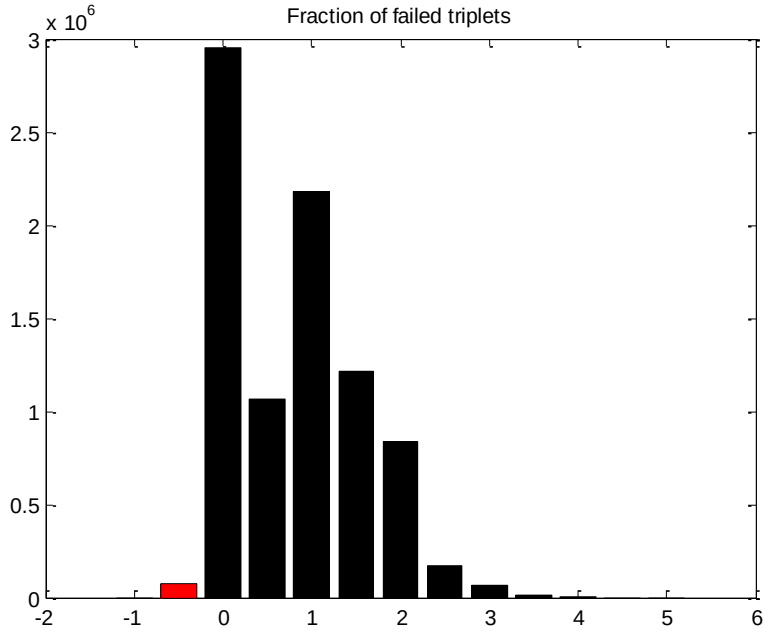



**Figure 7-22 Fraction of failed triplets**

The test was performed for all tag sets (all systems in the sample set). For each tag set the percentage of failed triplets over the entire set of triplets was measured. The result can be seen in
**Figure 7-23**. For the majority of tag sets the triangle inequality was not violated. For a small fraction of the tag sets the triangle inequality was violated to a small extend.

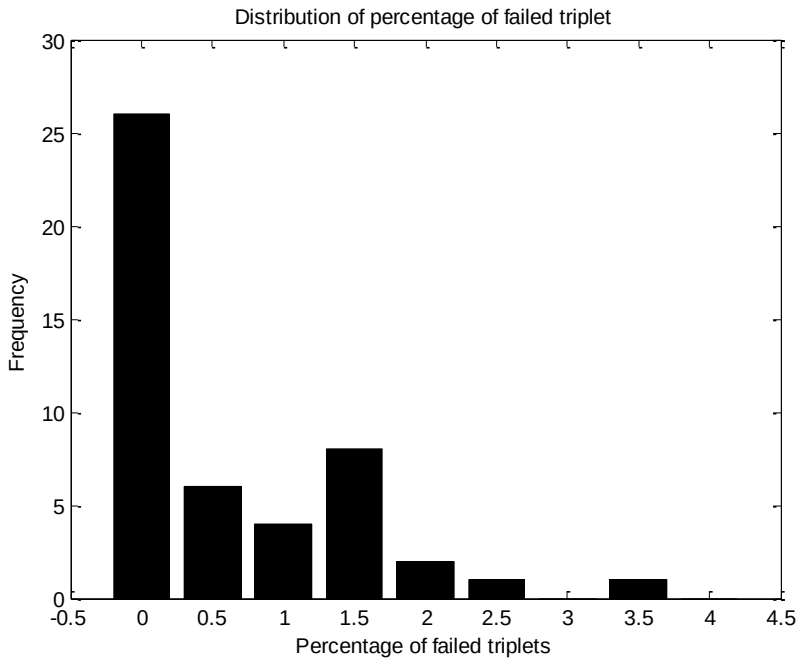



**Figure 7-23 Distribution of percentages of failed triplets**

Overall this performance is satisfactory. It indicates that the cost function that was defined here based on the associations of the semantics in the tags for the edit operation performs well when used in combination with the normalized edit distance.

## *7.6 Discussion and conclusions*

The test for the consistency of the characteristics of subsequences across the systems showed that the cutoff values for error and recovery sequences are not inconsistent. The visual inspection of the histograms (Figure 7-19 and Figure 7-20) and the uniformity tests, indicate that for both types of sequences, the subsequences tend to be coherent (right skewed histograms). These results encourage the collective use sequences from distributed systems for the parameterization of the segmentation algorithm foe newly installed systems. However this decision should be made considering that the test for uniformity was done based on the grouping of the cutoff values into cells, where the width of each cell is 32.625 and 43,667, for error and recovery sequence respectively. The width of the cell of the histogram defines the granularity at which differences in cutoff values are ignored. Rejecting the null hypothesis of uniformly distributed cutoff values, with small cell width is a stronger statement of consistency than with wide cell width because the differences in the cutoff values that are ignored are smaller.

Regarding the consistency of the association coefficients across distributed systems, the results show that only a small fraction 2.9% of associations of error semantics is consistently *strong* across the systems of the sample. This result might discourage the use of association coefficients obtained from other systems, in the matching operation of newly installed systems.

A striking observation is that even with a long observation period (sum of operating times of sample systems), there is shortage of data for statistical inference. For the error related semantics only 38.9% of the tag sets had adequate amount of

observations to perform the uniformity test. From these tag sets, 524 association sets were found, but only 7.4% had again an adequate number of observations to be tested.

It is clear that the uniformity test requires a large number of observations. Perhaps this demand can be mitigated with the use of "external" design data to complement the measure of associations between semantics. Such information can be retrieved from system design documentation that is describing the functional relation between components.

# Chapter 8

# 8 Knowledge discovery using transformed traces

So far the first phase of the transformation process, which is also the scope of the thesis, was presented. In this chapter the second phase of the process is described shortly. The aim of this chapter is to illustrate how analytical tools can be applied to the transformed sequences, which in turn can lead to knowledge discovery. Knowledge discovery is considered in the context of effective availability management. First the differences between using the methodology for parameterization on sample data and the filed application are described in 8.1. In section 8.2 knowledge discovery is demonstrated assuming the input is a transformed sequence. The chapter concludes in 8.3 with a comparative analysis between traditional failure data collection methods and the use of traces.

## *8.1 From parameterization to application*

The approach for applying the methodology consists of two main stages: a) Parameterization and b) Real time implementation. For each stage the steps of the transformation process are either relevant or not (Table 8-1):

1. Parameterization

At this stage the cutoff parameter for the segmentation algorithm is defined. The parameter is defined either for each system specifically or for groups of systems, depending on the result of the test for consistency of characteristics of subsequences.

2. Real time implementation

The methodology is applied in real time. Traces are processed as they are produced by the system. Once a subsequence is defined it is tagged, the cost function is updated and the tags (new and existing) are matched again.

| Operations is different stages of the application | | |
|---|---|---|
| | **Parameterization** | **Field implementation** |
| Data input | Sample sequence | Continuous data flow |
| Preprocessing | Yes | Yes |
| Segmentation | hCSM is estimated using the resampling method and adjusted to reduce the collision probability | hCSM is applied |
| Test for group characteristic cutoff value | Yes | No |
| Tagging | Yes | Yes |
| Tag matching | Yes | Yes |
| Test for group characteristic associations of semantic | Yes | No |
| Post Analysis (knowledge discovery) | No | Yes |

**Table 8-1 Overview of steps for two stages of application**

## 8.2 Availability management with traces

In this hypothetical scenario the focus is availability management using information from traces. The information is collected from transformed traces, now referred to as *tag type sequences (TTS)*. Here, an example of TTS is shown to demonstrate how it can be used for effective availability management. The methods are grouped under two tiers of knowledge generation.

1. The first tier uses directly the information from TTS. Direct measurements taken from TTS can provide information on the availability of individual systems and on the types of failures that cause the longest downtimes. The first tier supports also corrective maintenance.

2. The second tier uses the information from traces to support decision making in system design and system support. This tier is using the information from traces to feed modeling techniques that in turn can help improve the system design or the setup of the support system.

First the TTS is described in 8.2.1 then the tier 1 described in 8.2.2 and tier 2 in 8.2.3.28.3.3.

### 8.2.1 Tag type sequences

In the context of system resilience it is assumed that every system failure is followed by a system recovery. However, in a real case scenario the system may not recover by itself after each failure. For some failures a field engineer is needed to repair the system. In this example repairs are also considered to be recovery events.

The preprocessing and transformation of traces produces a TTS. The TTS consists of error tag types $E$ succeeded by recovery tag types $R$. The tag types $E$ and $R$ contain the semantic information describing the type of event. Each tag in the sequence is represented by the tuple$< E_i, t_k >$, where $i \in \mathbb{N}$, denotes the $ith$ error tag type occurring at time $t_k$where $k = \{0,1,2 \ldots N\}$, and N is the number of all events, and each recovery tag is represented by the tuple $< R_j, t_{k+1} >$, where $j \in \mathbb{N}$ denotes the $jth$ recovery tag type occurring at time $t_{k+1} > t_k$. The observations start at the time of the first error tag and $t_0 = 0$ (synchronous).

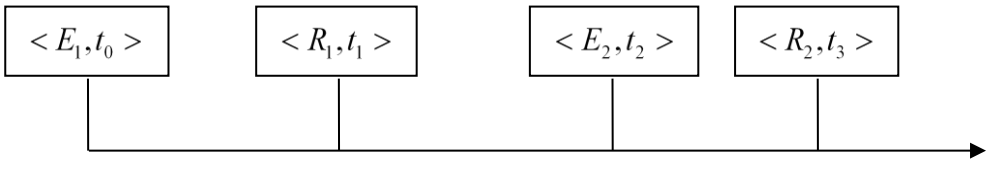



**Figure 8-1 Tag type sequence**

In Figure 8-1, two error tag types can be seen. Each is followed by a different recovery tag type. During the times of $t_0 - t_1$ and $t_2 - t_3$ the system is unavailable.

### 8.2.2 Tier 1: Measurement and corrective maintenance

To measure the system operational availability the information from the tag type sequence is used directly to calculate the system down time:

$$Down\ Time = \sum_{z=0}^{N/2-1} t_{2k+1} - t_{2k}$$

and the total operating time of the system from time $t_0$ to the current point in time $t_c$

$$Total\ Operating\ Time = t_c - t_0$$

Then the operating availability is:

$$Operating\ Availability = \frac{Total\ Operating\ Time - Down\ Time}{Total\ Operating\ Time}$$

For the purposes of corrective maintenance the TTS can be used as following: at the occurrence of a new error event, the error tag is either categorized into a known tag type or it defines a new tag type. In the first case corrective maintenance can proceed (assuming system does not recover by itself) based on the knowledge that is already available for the existing tag type. In the second case diagnostics have to be performed before maintenance can begin. Once maintenance is completed and the correction has been verified, the information regarding the error is added to the newly defined error tag type.

### 8.2.3 Tier 2: availability modeling and association rules

#### 8.2.3.1 Availability modeling: identifying bottlenecks

Stochastic modeling is used for availability analysis. Availability analysis provides computations of *point availability,* i.e. the probability that the system will be in up-state at a time *t,* and *steady state availability* or *limiting availability* i.e. the availability of the system in the long run. Measurements taken from traces can be used as input for availability models.

To illustrate this, an example is provided: the failure process of a complex system is modeled as a composite of the failure processes of single failure types. Each failure type together with its associated recoveries is modeled by a single *failure type process*. When all *single failure type processes* are superimposed they compose the system failure process (Figure 8-2). It is assumed that failures are occurring independently from each other, that they don't occur at the same time and that after every recovery the system is as good as new.

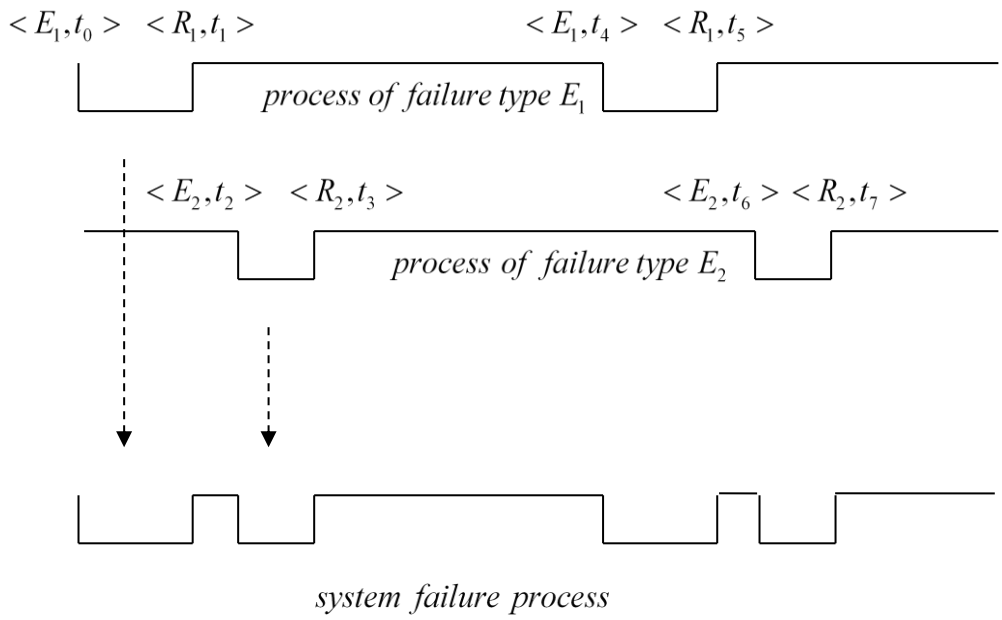



**Figure 8-2 Composing system failure process by superimposing single failure type processes**

The system failure process describes essentially a system made out of multiple components put in series (Figure 8-3). This configuration is based on the principle that when one component fails the system fails. Here each block in Figure 8-3 represents one failure type rather than a physical component. The analogy remains: when any type of failure occurs the system is down.

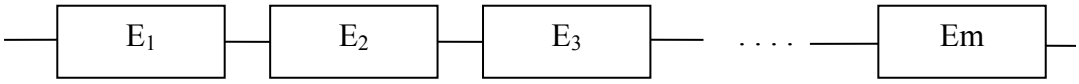



**Figure 8-3 Reliability diagram of a multi component series configuration**

#### *8.2.3.1.1 Simple failure type process defined*

A *simple failure type* process according to the above description is defined analytically as follows:

When a failure event of failure type $E_i$ occurs at time $t_k$ the system becomes unavailable for a period of time $U_{k+1} = t_{2k+1} - t_{2k}$, until it recovers by the recovery event $< R_j, t_{k+1} >$ . After recovery, a period $A_{k+1} = t_{2k+2} - t_{2k+1}$ follows where the system is operational. The process continues with the same alternation. The alternation of the intervals $A_{k+1}$ and $U_{k+1}$ form an alternating renewal process [Ave98].

The length of intervals $A_{k+1}$ and $U_{k+1}$ are modeled as random variables, with distributions $F$ and $G$ respectively. Mean Time to Failure $\mu_F$ and Mean Time to Recovery $\mu_G$ are the means of these distributions.

Given a series of operational and recovery intervals $A_1, U_1, A_2, U_2, \ldots$ the following two variables are introduced [Ave98]:

Time to the n$^{th}$ failure: $S_n = A_1 + \sum_{k=1}^{n-1}(U_k + A_{k+1}), n \in N$, **( 8-1)**

and the time to the completion of the $nth$ recovery: $S_n^o = \sum_{k=1}^{n}(A_k + U_k), n \in N$ **( 8-2)**

The sequence given by $S_n^o$ forms an ordinary renewal process $N^0$ with distribution function $H^{o(n)}$, where $H^{o(n)} = (F * G)^{*n}$, and (n) denotes the n-fold convolution $(*)$ of the distribution $H^0$ process [Ave98]. The renewal function $M^0$ of $S_n^o$ is:

$M^0(t) = \sum_{n=1}^{\infty} H^{0(n)}(t)$ **( 8-3)**

Given the above, the point estimate for the availability $\mathrm{Av_i(t)}$ of the system in relation to failure type $E_i$ is:

$\mathrm{Av}_i(t) = \overline{F}_i(t) + \overline{F}_i * M_i^o(t)$ **( 8-4),**
where $\overline{F}_i = 1 - F_i$

and the point estimate of unavailability $\overline{\mathrm{Av_i}}(t)$ because of failure type $E_i$ is:
$\overline{\mathrm{Av}}_i = 1 - \mathrm{Av}_i(t) = F_i(t) - \overline{F}_i * M_i^o(t)$ **( 8-5)**

The limiting availability $\mathrm{A_i}$ is defined as:

$\mathrm{Av}_i = \lim_{t\to\infty} \mathrm{Av}_i(t) = \frac{\mu_{iF}}{\mu_{iF} + \mu_{iG}}$ **( 8-6)**

and the **limiting unavailability** caused by failure type $E_i$ : $\overline{\mathrm{Av}}_i = 1 - \mathrm{Av}_i$ **( 8-7)**

#### *8.2.3.1.2 System failure process defined*

Having modeled the simple failure processes, the composite system failure process can be obtained by superimposing the $l$ sequences $S_{ln}^o$ of the failure types $E_i$.

The point availability of the system at time $t$ is given by:

$\mathrm{Av}_{\mathrm{System}}(t) = \prod_{i=1}^{m} \mathrm{Av}_i(t)$, **for** $i = 1, 2, 3 \ldots m$ **( 8-8)**

and the limiting availability for the system

$\lim_{t\to\infty} \mathrm{Av}_{\mathrm{System}}(t) = \prod_{i=1}^{m} \frac{\mu_{iF}}{\mu_{iF} + \mu_{iG}}$ **( 8-9)**

From the above formulas, the system limiting availability provides the information whether the system performs as good as expected. If the system availability is below the desired level, the availability bottlenecks can be identified by the ordering of the point estimates or the limiting unavailability of failure types.

#### *8.2.3.1.3 Measurements from the TTS are used for parameter estimation of availability modeling*

The distributions $F$ and $G$ are obtained from the TTS by measuring the time of occurrence of failure types $E_i$ and their associated recoveries $R_j$. Parameters for these distributions are estimated based on these measurements.

For a failure type $E_i$ the first time to failure is the interval $A_{i1} = t_{i0} - t_0$, where $t_0 = 0$ is the start of the observation and $t_{i0} > 0$ (if first failure is $E_i$, then $t_{i0} = t_0 = 0$). For failure type $E_i$ the first time to recovery is $U_{i1} = t_{i1} - t_{i0}$ where $t_{i1}$ is the time of the subsequent recovery event $R_i$. For the *kth* time to failure we

$A_{ik} = t_{ik} - t_{ik-1}$ **( 8-10)**

and for *kth* time recovery:

$U_{ik} = t_{ik+1} - t_{ik}$ **( 8-11)**

The above measurements are taken for all occurrences of each error tag type and their associated recovery tag types found in the label sequence.

Using this measurement the limiting unavailability for each error tag type can be estimated. By ranking the tag types in descending order of limiting unavailability, a list of availability bottlenecks is obtained, prioritized from the most severe to the least severe. Improvement actions can be taken after analyzing the list top down.

### 8.2.3.2 Association rules

The information from availability analysis focuses on the availability bottlenecks. Since the modeling approach is failure centric i.e. one failure process defined for a failure type, the information collected by now is pointing out to the most damaging failure types.

To improve availability system designers want to examine two options:

1. reduce the frequency of failures
2. enhance system resilience

To support the above decision, semantic information on the types of failures and recoveries is obtained from the label sequence with the discovery of association rules.

Association rules in temporal data sequences describe the relationship between two or more events. In the case of failure and recovery events in traces the association rules describe causal relationships between the two events [Rod02]. Using sequential or temporal data mining tools it is possible to detect and measure the relationship between an error tag type $E_1$ and its associated recovery tag types $R_j$.

Association rules on error and recovery sag types can help complement the information regarding the bottlenecks of availability. Association rules provide qualitative information on the availability performance. For the tag type $E_1$, mining the TTS for association rules can reveal the recovery tag types that are mostly related to it e.g. $\{E_1\} \Rightarrow \{R_1, R_4\}$.

With the information about the association rules the engineering analysis can be directed to the semantics information of the tag types. A root cause analysis can provide more details on the nature of the error and the recovery event. The gain here is that the root cause analysis, which is expensive and time consuming, is performed on already prioritized problems.

## 8.3 Availability management & decision making: past, present and future

The computerization of systems did not only come with new challenges it came also with new abilities in information sharing. In contrast to traditional systems with strong mechanical and electrical designs, these systems can record, collect, store and share data about the status of their components and the overall state. The easiness with which this information can be recorded and shared has a strong potential in transforming many aspects of availability and reliability management of professional systems.

Both availability and reliability management depend on information. Information is required about the operational status of systems in the field, for diagnostics and maintenance planning and for spare part inventory management. Availability and reliability analysis techniques that have been developed for a range of problems such as, design decisions, maintenance policies, spare parts inventories, rely on data input collected from the field. Without it, even the most sophisticated analytical method will become just academic exercises. In this section the limitations of traditional data collection techniques (8.3.1) and their effect on the effectiveness of availability management (8.3.2) will be presented. To illustrate that effect, four decision making areas that relate to availability management are examined:

- **Corrective maintenance**: Restore operation to the system after failure
- **Preventive maintenance**: Reduce the frequency of system failures by replacing critical components based on information about their reliability
- **Resource management**: Management of spare parts inventories, logistics and maintenance staff Product creation
  Design and build high availability into systems
- **Performance indicators**: Measurements of availability system performance

Then in 8.3.3 it will be shown how traces provide advantages for data recording and collection, and how these advantages affect the same areas of decision making (8.3.4).

### 8.3.1 Conventional methods for failure data detection and reporting

In this section we will show how widely used current practices, used by manufacturers for capturing and reporting field failure data miss to provide the information required for effective availability management.

#### 8.3.1.1 Data via help desk

It is not unusual that a system failure experienced at the customer's site is reported to the help desk of the service provider. This practice has several drawbacks:

- The response for corrective maintenance is delayed by the handling of the call.
- The information that is collected this way contains as little as the time of the call, name of the customer, and a vague description of the problem [Pet03].

- For the purposes of analysis this data collection introduces left truncated data, as the exact moment of failure is most likely unknown [Bha03].

#### 8.3.1.2 Data from field reports

The help desk will try to solve the problem remotely, given that its severity is low by giving instructions to the customer to overcome the problem. If this is not successful a field service engineer (FSE) is sent to the site. More information on the failure will become available only when the FSE is on site and after the diagnosis on the system is complete. It may be the case that under time pressure to restore the system, the FSE will choose to replace parts in trial and error fashion, to bring the system back in operational state as quickly as possible.

The information on the completion of the corrective maintenance is added to the failure report, containing for example repair activity description, parts used, hours spent for the completion of the maintenance. This data reporting method has severe impact on the accuracy and availability of failure information and can obstruct or mislead further use [Pet03] [Bla98].Often the time scale used is calendar time, which puts limitations on the use of the data and values of covariates are entirely missing [Law92]. Reports of this type may be filled by staff of different skills, in different language, in different locations. This suggests that extracting information from them can be an elaborate, time consuming process.

Warranty data are often a source of failure information for manufacturers. They usually are produced by the FSE. They have however some limitations that relate to the length of the warranty period. Because the data are of interest for the period of time the product is under warranty, data beyond that point are not recorded. The result is that organizations are left with truncated and right-censored data sets [Bha03]. Another issue with warranty data is the time for the data to become available. From the moment of the warranty claim until the data is available time will pass due to administrative and processing issues [Law92].

#### 8.3.1.3 Data from surveys/cross sectional samples

Surveys and cross sectional sampling are performed by external parties or the manufacturers themselves to collects system failure information. They are often carried out as a response to perceived problems in the field. Some issues with the data that have been collected this way is that the data on the surveyed products are response related, and they may be heavily censored and truncated [Law92].

### 8.3.2 Consequences of conventional methods on availability management

Conventional practices in failure reporting involve extensive manual labor and human interference, causing inconsistencies in the data. They are lengthy and time consuming practices. They also fail to provide a complete overview on the product's performance in the field due to their strong dependency on activities such as corrective maintenance. Conventional failure reporting practices fail to support availability management in:

**1.** Corrective maintenance

The data collected via customer calls has little informative value. It has no technical information, since the customer is not capable of commenting more on the failure than

the observations that can be made on the product's state. The result is that the FSE has to perform full diagnosis, detection and isolation of the fault on arrival on the site. For the complete diagnosis of the product, testing tools may be required that are not often available at the first visit. In this case, a second visit is required to complete diagnosis. Only when the diagnosis is complete and the fault has been located, the repair or replacement can begin. The tools and/or parts needed for the repair/replacement have to be ordered and shipped to the site. The above prolong the duration of system down time.

**2.** Maintenance Policies

Poor data input results in limited options for maintenance policies. Many companies would like to make use of more effective maintenance methods but are hindered by the lack of appropriate information input [End01] [Bla98].

**3.** Resource management

Lack of or inaccurate information on demand of parts can increase inventory levels and subsequently cost of stocks. Inaccurate demand information can result to delays in the delivery of parts due to bad transportation planning or facility misallocation and staff unavailability, which would lead to service delays [Cho04].

**4.** Product Creation Process

Censoring of data (right and left) is one of the main issues that come with current failure data recording practices. Dealing with censored data in reliability analysis requires the use of appropriate methods [Ans89], but it is doubtful whether estimates deriving from such data can be accurate [Coi86]. The analysis of life data requires knowledge on failure as well as on survival data (product without failures) otherwise reliability estimates will be pessimistic [Coi86]. Current methods however, do not keep track of systems without failures.

Due to absence of failure data, engineers often make too simplistic assumptions on the failure rates of electronic components when predicting product reliability in the development process. Moreover, the use of standard sources of failure rate as the MIL-HDBK 217 handbook can lead to great inaccuracies in reliability prediction because they neglect the effect of environmental factors [Bla98] [Woo94]. Possibly one of the biggest shortcomings in the current practices of recording failure data is the absence of information on covariates, environmental or use conditions. Mixing data from various sources does increase the level of uncertainty on parameter estimation of probabilistic models.

Many current failure recording methods are not capable of keeping track of software failures since these cannot be handled by FSE or the help desk. The delay in making field failure data available for analysis, is an obstacle in their use for detecting failure trends quickly and take actions on time to remedy design flaws of existing products.

**5.** Performance indicators

Organizations miss to provide their customers with estimates of LCC and TCO, a factor that becomes increasingly important in purchasing decisions. They also miss to provide their customers with feedback about their products' performance in availability. Such reports would prove their commitment for continuous product evaluation and improvement.

### 8.3.3 System failure information from traces

Managing system availability throughout the product life cycle requires the appropriate data to be captured and transformed to information and passed on to the right parties at the right time and in the right format. To achieve an effective end-to-end solution, automation needs to be incorporated at all stages of the *information system,* i.e. data capture, collection, storage, processing and distribution.

Traces have great potential for availability management of professional systems because:

- Its implementation is designed into the product and assures data availability and effective coverage of all critical components
- Being in digital form, these data can easily be sent via remote connectivity to any location allowing remote, close to real-time, monitoring capabilities
- No human intervention assures such high levels of data consistency
- The digital form of the data allows their integration with IT and machine processing applications, making data analysis and information available and fast
- Its relatively inexpensive implementation allows the monitoring of fleets of products in the field

### 8.3.4 Advantages of traces for availability management

The advantages that come with traces can be found in all areas relevant to availability management:

1. Corrective maintenance

Traces can provide diagnostic support to FSEs. Semantic information in traces can indicate the location of the fault, This information is available immediately after system failure and can reduce corrective maintenance down time, by guiding the FSE in locating and isolating the fault and by making sure that the right tools and testing equipment are on site the same time as the FSE at the first visit. In combination with parts tracking technologies, the needed spare parts will be requested at the time of failure which can reduce further delays due to logistics.

2. Preventive Maintenance

Failure data components are collected accurately and timely, which can help in the planning of preventive and predictive maintenance activities. Moreover, digital components can now be included in preventive maintenance planning. For software components for which failure intensity is measured high, bug fixes can be provided remotely.

3. Resource Management

Real time data availability can provide accurate and timely forecasting information for spare part inventory management and manufacturing. It can also provide similar input for logistics and maintenance staff planning.

4. Product Creation Process

Field data on product performance from the moment of installation until decommissioning will provide a valuable input for product evaluation. Continuous

monitoring of failures in the field can detect timely availability trends that might suggest the need for design modifications. Additional information on product use can help in the improvement of the design, by adding redundancy or increasing resilience to overcome design shortcomings.

5. Performance indicators

Warranty estimates will be based on actual field performance and usage. LCC and TCO estimation can be done on the basis of accurate and complete use of spare parts, maintenance activities, down times etc.

#### 8.3.4.1 Improving the logging mechanism

As the observations on systems are made based on the analysis of traces, new requirements for the logging mechanism arise. New areas of interest need to be covered to gain more insight in the role of components on the system's availability. The quality of traces needs to be improved, by trying to record traces in more compact (close temporal proximity) manner to increase the effectiveness of the mining algorithms. The improvement on the logging mechanism goes hand in hand with the need for more information to provide new/better insight into the system's behavior.

# Chapter 9

# 9 Conclusions and recommendations for future research

This chapter concludes the thesis with the discussion on the objectives of the research, the validity, the scientific and practical contribution and the recommendations for future research. First the objectives of the research are discussed in 9.1. Then in 9.2 the validity and reliability of the proposed methodology is discussed. In 9.3 the scientific and practical contributions are presented and in 9.4 some reflection on the work around this research are laid out. Finally section 9.5 provides some recommendations for future research.

## *9.1 Research Objectives*

In this section the objectives that were set in 1.6.2 are presented and discussed separately.

The main objective of the research is the reduction of the size of traces without the loss of any information regarding the events that are relevant for the availability management of systems i.e. failures and recoveries. The data reduction should also lead to a new data representation that is suitable for specific analytical tools used for system availability analysis. The objective is discussed in its parts:

**Reduction of the size of traces without the loss of relevant information**
Data reduction is performed throughout the preprocessing and the transformation stage. During preprocessing the pps found in the sequence of traces is removed. A pps can consist of thousands of entries over a period of hundreds of hours of operating time. With the removal of the pps the sequence can be processed effectively.

The transformation stage is where the data size reduction is achieved formally. The basis for the reduction of the size of data is to reduce the amount of distinctive data points required to convey the same information. Before the transformation the "raw" sequence consists of multiple instances of traces representing the occurrence of one or more physical events. Each trace, being a data point, carries a piece of the information about the physical event it represents. After the transformation stage the tag types are the new data representations of physical events and the sequence is referred to as tag type sequence. The tag type sequence conveys the same amount of information as the original sequence with fewer representations. In the case study it is shown that the data compression ratio can exceed the order of 0,01 i.e. 100 times fewer data points are needed to convey the same amount of information as the original sequence (see case study chapter 6). With the data size reduced, not only is it possible to use the tag type sequence in analytical methods, but it becomes easier for engineers to use the information in traces for diagnostic purposes. Throughout the transformation the temporal and the semantic information is not lost. The temporal location of the physical event and the semantics are inserted into the tag.

**Point representation of physical events**
The tag type sequence consists of point representations of physical event instances. Through the transformation stage, with the segmentation the traces are clustered into subsequences, each representing an instance of a physical event and with the tagging the subsequence becomes point representations. This method of representation is not only suitable to specific analytical methods (see chapter 8), it also allows effective manual inspection of the data because of its reduced size and its organization into event types.

The above two objectives are met under the conditions:

**The methodology is generic and does not require system specific information**
The methodology for the transformation of traces does not depend on system specific information, such as design documentation. Also complementary information such as expert domain knowledge was not used. As it was shown in 3.1, domain expert can introduce uncertainty in their interpretation of traces, and are therefore not a dependable source of information. To avoid external data dependencies the methodology is using generic domain knowledge and the appropriate interpretation of the data structures found in the sequence. The generic domain knowledge comes from the system engineering domain and suggests that complex system have modular design. Based on that, the temporal proximities of traces and the association between semantics in the sequence are used to develop the methodology. Such information is inherent in long sequences of traces of any system that has been designed according to the modular principle. The information that is needed for the transformation of the raw traces of a system is taken from a sequence of that system. Specifically the sampled sequence is used to set the values of the cutoff parameter for the segmentation (chapter 4) and the values of the cost function for the tag matching operation (chapter 5). This approach allows the methodology to be applied on any type of professional system that has a modular design.

**The methodology can deal with variation inherent in the traces that might affect the transformation results**

Variation is present in traces. As it was shown in the fault injection experimentation (3.2) and the visual representation technique (3.3), multiple instances of the same type of physical can result to several variant subsequences. The variation is twofold. Firstly it is found in the temporal distance between consecutive traces in each variant subsequence. Secondly the semantics can vary in the type and number of occurrences in each variant subsequence. Nevertheless the variant subsequences seem to contain semantics from a single pool. To handle these two types of variation three methods where used:

a. The cutoff parameter used for segmentation process is robustified during the segmentation of the sample sequence, by using the resampling technique (4.2.3). The resampling technique simulates the presence of variation in the temporal distance between consecutive traces in the subsequence. The amount of simulated variation can be controlled.
b. Redundant occurrences of semantics in a subsequence are eliminated with the tagging (5.1). The tagging also reduces variation by ordering the semantics lexicographically. Both operations do not have a negative impact on the information. As it was shown in the fault injection experimentation, multiple occurrences of the same semantic and the order of the semantics is an artifact of the logging mechanism.
c. The tag matching operation is capable of dealing with variation found in the lengths of tags (5.2). In order to find matches between tags, the semantics of these tags are compared. However tags are not necessarily of the same length. To avoid penalizing differences between the lengths of tags a normalized dissimilarity measure is used. To use this measure effectively a cost function is defined that allows the distance measure to meet the metric properties. The proposed cost function also allows the comparison of tags for their (dis)similarities, according to the underlying engineering dependencies between components incorporating that way the system design information in the tag matching process.

**The methodology has to be applicable for close to real-time data processing**

The methodology in this thesis is designed to process data in sequential manner. In field applications traces can potentially be transformed as they are produced by the logging mechanism in a close-to real time manner. The segmentation algorithm can process traces as they are produced and define subsequences. The subsequences are transformed then into tags. The relative frequencies and the association coefficients can be updated also a close to real time manner. The cost function is updated and the tags are being matched into tag types (see section 8.1).

**The methodology has to exploit the number of available systems to increase its efficiency**

The methodology can benefit from the use of traces from multiple identical systems that are geographically distributed. The efficiency of the methodology can increase if the learning from a group of systems of the same type can be generalized for that system type, and applied to any newly installed system. Such a generalization can decrease the period of time that is needed to until the traces of the newly installed system can be utilized because it allows the use of the same parameter values for the same system types. To enable such generalizations, the methodology proposes formal tests for investigating whether the subsequences collected from distributed systems show consistency in their structural characteristics (chapter 6).

## 9.2 Research validity and reliability

In this section the validity, internal and external, and the reliability of the research are discussed.

### 9.2.1 Internal validity

Internal validity is the extent to which systemic error has been minimized in defining and formulating the causal relationship between physical events such as system errors and recoveries and the subsequences that represent them. The understanding of this relationship affects the methods that are developed in the transformation methodology.

The information was collected from various sources and using different methods. The proposed methodology is based on an extended literature review of the domain and on a rigorous exploratory phase, which led to the good understanding of the mechanics behind the formation of data structures found in long sequences of traces. During the exploratory phase the observations were made on the structure of sequences of traces from three different perspectives, the input of domain experts, experimental setup, and graphical analysis. There has been maturation in familiarity with traces during these three phases that spanned over a period of 12 months.

A large data sample was selected for observation and analysis. The graphical analysis and the case study are making use of a data sample that was selected for the completeness of the trace sequences and the homogeneity of the systems that produced that. The identical systems share the same system design i.e. system architecture and the same logging mechanism i.e. same semantics. This similarity allows comparisons of the observations on the data structures across the sequences. However, even though the systems where operating in the same application field, they were from different geographical locations and of different lengths of operating times. These differences introduce the variation in the sequences that allow observations to vary enough to provide a wide view on the spectrum of what is possible in the formation of subsequences.

To avoid the bias towards specific forms of data structure, the experiment of fault injection, faults where injected in software and hardware components, and in different layers of the architecture. This design allowed a wide coverage of the system's architecture and provided wide range of observations on the structure of subsequences in relation to the location of the root cause.

### 9.2.2 External Validity

External validity refers to the extent to which findings and methods can be generalized. The objective of this thesis is to provide a generic methodology for the reduction of the data size in traces. A threat to external validity is the sample specific features that affect strongly the finding and limit generalization.

To avoid adopting features that are specific to the sample used in the research, all findings of the exploratory phase are abstracted to a level where the relation with the specifics of the sample is weak but the general features of interest remain strong. Though the observations are made on data produced by a specific type of professional system, the key information that describes the features of interest refers to generic

characteristics, such as close or distant temporal proximity, or strong or weak associations.

In the end of the exploratory study (chapter 3), 4 conjectures are stated on which the methodology is based. All 4 conjectures are statements on abstract features of the data structures found in traces and they apply on a wide range of professional systems. These conjectures are set under assumptions of known and well established engineering principles i.e. modular system design.

The method of data collection also helps the generalization of findings. The experimental setup and the graphical analysis of the sample set, provided a two dimensional perspective onto the data structures of traces. In the latter case an in depth view into the interaction of the system's design and the formation of subsequences was gained and in the former case a wide view across multiple systems allowed the comparison of structures and the identification of commonalities and differences. Moreover the sequences in the sample set originate from an X-ray scanner that is considered to be complex system. Data from a complex system are more likely to contain a big variety of data structures to make observations on.

Overall the proposed methodology was developed with generalization in mind. It includes methods to help characterize the data set of a new application before implementation. The methodology proposes to use the entire framework, from preprocessing and exploration, to tag matching in any new situation. Also the methods that reply on parameterization require the fitting of parameter values on the new data set.

### 9.2.3 Reliability

Reliability in this research is understood as the dependability on the proposed methodology to return credible results when applied. The reliability of the proposed methodology can be credited to two aspects of this research. The first aspect is the ability to understand and describe the difficulties that need to be overcome to meet the objective i.e. data reduction. The second aspect of the research is the effectiveness of the methods that can be used to overcome these difficulties.

The first aspect was addressed by the discussion over internal validity. Avoiding the bias in the description of the problem is part of a reliable problem description. From the description of the procedure followed in the fault injection experiment and in the graphical analysis, care has been taken so that observations are credible. In the experiment, the system type that was chosen is identical to the type of the systems used in the sample data set. Between fault injections the system is reset to assure independency of the experiments. In the graphical analysis, observations made in on sequence are compared with others. The features that are observed are considered credible, for example the presence of pps across different systems or the close temporal proximity of traces of the same subsequence.

The reliability of the proposed methodology is derived from different levels. On high level the methodology is designed around few clear guidelines for overcoming the identified difficulties. In both segmentation and tag matching the methodology strives for compact clusters. Having simple and clear rules helps to choose methods that can best the requirements.

At a lower level, where distinctive methods are proposed, proofs or evidence are provided where necessary for the methods:

1. The discovery of pps in long sequences of traces (3.4). A computational analysis is provided that proves that the proposed method for detecting pps is more effective than current state of the art.
2. Choosing the value of the thinning probability (chapter 2). A computation analysis is provided to help chose the value of the thinning probability that will yield the best results in terms of robustification of the cutoff parameter.
3. The normalized edit distance is more suitable for use with traces than non-normalized (chapter 5). An example is provided that shows how the normalized edit distance performs better than the Levenshtein edit distance in grouping together similar tags.
4. Cost function meets all metric criteria (chapter 5). An analytical proof is given that shows that all criteria of a metric are met by the proposed cost function.

Decision making aides and tests are provided where necessary to guide the implementation of the method and assess results. Unsupervised data mining methods are used in the transformation process for the steps of segmentation and matching. For unsupervised methods there are no external data to either train the algorithms or validate the findings. For this reason the methods are fitted with rules that can guide the implementation. For the segmentation of the sequence the implementation is guided by the hCSM and the tag matching is guided by the silhouette stopping rule. In addition an "internal validation criterion" i.e. a hypothesis test, is used to assess the plausibility of the segmentation and for the tag matching operation the performance of the cost function is assessed by the "looseness equality test".

The reliability of the methodology also derives from the detailed description used throughout the thesis allowing the reader to critically assess it. A case study is used where the implementation of the proposed methodology is demonstrated.

Particularly useful for the reliability of the methodology is test for unimodality for the characteristics of subsequences across systems. This test does not only provide the means to make the implementation of the methodology on a group of systems more efficient but is a way to verify that when the methodology is applied on different systems the results have a degree of agreement that can be tested formally. The agreement of the results is proof of the methodology has repeatability when applied on different data sets.

## *9.3 Contribution of thesis*

The contribution of this thesis is twofold, scientific and practical. The scientific contribution is found in the new methods that are presented in the research and that been developed to address specific problems found in the data structures of traces. The practical contribution is found in the entirety of the methodology that is the first end to end process that describes the utilization of traces for the availability management of systems.

### 9.3.1 Scientific contribution

The scientific contributions are the new and some improved methods proposed thought the thesis, for processing event based data sequences. In chapter 3 in Section

3.1 the multi class membership classification scheme together with the use Shannon's entropy theorem, constitute a novel method for assessing the uncertainty of our knowledge on relationships between events or objects and concepts. The method is particularly interesting because it allows updating of the uncertainly measure with new information to obtain a new measurement of the level of our understanding.

In section 3.3 an efficient visualization method was presented that is suitable for fast visual exploration of event based data. The method does not require rigorous data mining algorithms to detect associations. It uses a simple mapping of associations between data types in the form of a bitmap. The method is not precise because it is uses arbitrarily defined intervals to segment the sequence but it is easy to adjust and to deploy for fast data exploration. Nevertheless the method can be improved to use data mining segmentation methods.

In the same chapter in section 3.4 the pps problem gave the opportunity to develop a more efficient detection method of pps in a data sequence that requires only one pass through the data sequence, as opposed to several in previously proposed method, to make the detection of pps possible. This was achieved with the definition of a hypothesis test that is based on a mixed Erlang distribution.

In chapter 4 the hybrid cluster separation measure is an improvement of the existing separation measure, because it prevents the over fitting on the data. The hybrid separation measure was defined using an existing separation measure and a resampling method. The result was a method that accounts for variation in the event based data sequence.

In chapter 5 a novel cost function was defined based on the measure of association between semantics. The cost function fulfills all criteria of a metric and makes it suitable for measuring the similarities between ordered subsequences of traces. In the context of system generated traces it is the first cost function for comparing semantics.

In chapters 6 the concept of characteristics of the structures of subsequences is defined. The definition uses the temporal and the semantic aspect of the subsequences to specify the structural characteristics. The definition allows the comparison of subsequences obtained from distributed systems to be compared for similarities in their structure.

### 9.3.2 Practical contribution

The practical contribution of the thesis has been elaborated extensively in chapter 8. The proposed methodology is the first end-to–end process that enables the transformation of raw traces to sequences of point representation of physical failure and recoveries. The transformed sequence enables the use of analytical tools, such as availability modeling and data mining techniques to discover interesting system and component performance characteristics. The methodology is designed with application in mind. The data are processed in sequential manner. This widens the spectrum of applications allowing not only retrospective analysis but close to real time application to support operational activities of an organization. Finally the methodology allows the use of numerous distributed sources of traces to increase the efficiency of the application.

## 9.4 Reflection on work

Even though measures were taken to increase the reliability of the methodology for producing valid results, there are some methods that were not used, which can contribute to this objective:

### 9.4.1 Feedback loops

The use of unsupervised data mining methods in the methodology might leave some doubt on the correctness of the results. Since there are no external data to validate the results, the correctness relies on the use of the guidance aides (hCSM, silhouette value), and the tests (internal validation criterion, looseness equality test) to provide confidence. Another valuable element in this methodology would be the use of feedback loops particularly when the methodology is used in operational conditions in the field. In their simplest form these feedback loops would require expert engineers to assess the results of the methods and recommend corrections where necessary. For example one feedback loop can inform how the cutoff value performs in respect to the risks of truncation and collision and make adjustments if necessary. Another feedback loop can inform on the correctness of clustering tags into tag type. In this case an engineer can inspect the clustering results and make adjustments to the cost function to either force separation or groupings of tags. Such feedback loops were not described in this thesis, but undoubtedly they will be valuable assets for increasing reliability of the methodology.

### 9.4.2 Use simulated data to verify the methodology.

Since the clustering operations used in the methodology fall under the category of unsupervised learning, it is hard to validate the results without having an exact description of the expected outcome. This shortcoming of unsupervised machine learning, cold have been mitigated, if the methodology would have been validated using also simulated sequences with known number of subsequences and known structure of these subsequences.

### 9.4.3 Assess the performance of real time applications

The proposed methodology is designed with real-time applications in mind. However the main focus in thesis is the parameterization of the algorithms before these are put to use in the field. In the case study too, the methodology is applied on the sample data on off-line mode. To have a full assessment of the methodology, this should be tested in an on-line setting. Such a test would give information on how well the parameterized algorithms perform on real time data in terms of correctness and efficiency. The learning from this test can be used for recommendations to improve the proposed methodology.

## 9.5 Recommendations for future research

The problem of reducing the size of traces and transform them into sequences of point representations allowed the identification of some new interesting aspects of the domain that can be explored. Here some of the interesting topics are listed.

### 9.5.1 Designing the logging mechanism

The relationship between the logging mechanism and traces is important. The amount of sensing points that the mechanism has is decisive for the traces. Anything that is not covered by the sensing network remains invisible for later analysis. In addition to

that the logging mechanism has a strong effect on the formation of the subsequences. An interesting research subject is to provide a framework for designing an effective logging mechanism with a dedicated logging format and performance that can cover all critical components and can provide data sequences that can be processed easily. Such a framework can make use of system Failure Mode and Effects Analysis methods as input for designing an effective sensing network. Data recording protocols could provide data sequences with compacts subsequences and reduced number of redundant entries.

### 9.5.2 Integration with information systems for real time applications

An implementation of the methodology proposed here would be part of an integrated information system that would cover the entire area from the front-end system data recording and collection methods to the back-end of information provision for supporting decision making for system availability management. The design of such a system is challenging. It would make use of state of the art information system technology and business requirements to provide an end-to-end solution. The proposed methodology would be one of the core engines in such a system.

### 9.5.3 Define and refine the range analytical methods that can be applied on information from traces.

The focus for analysis using information from traces was put in this thesis on availability modeling and discovery of association rules. These two methods where used as examples to illustrate the use of traces in decision making on availability management. A review on the area of availability management problems can reveal a wide range of analytical problems that can benefit from traces.

# References


[Air08] Airport International, July 2008, Heathrow T5 Losing 900 Bags a Day, Airport News

[Ana95] Anand S.S., Bell. D.A., Hughes J.G., 1995, The role of domain knowledge in data mining, 4th Int'l. ACM Conf. on Information and Knowledge Management, pp. 37-43

[And95] Anderson R., 1995, Software system for automatic parameter logging on Philips SL20 linear accelerator, Medical and Biological Engineering and Computing, vol. 33, pp. 220-222

[Ans89] Ansell J.I., Phillips M.J., 1989, Practical problems in the statistical analysis of reliability data, Applied Statistics, vol. 38, no. 2, pp. 205-247

[Ant01] Antunes C.M., 2001, Temporal data mining: An overview, presented at the Workshop on Temporal Data Mining With the Int. Conf. Knowledge Discovery and Data Mining,

[Arl90] Arlat J., Aguera M., Amat L., Crouzet Y., Fabre JC., Laprie JC., Martins E., Powell D., 1990, Fault Injection for Dependability Validation: a Methodology and Some Applications. IEEE Trans. Software Engineering, vol. 2, pp. 166-182

[Ars00] Arslan A., Egecioglu Ö., Efficient algorithms for normalized edit distance, Journal of discrete algorithms, vol. 0, no. 0, pp. 1-18

[Asi98] Asiedu Y., Gu P., 1998, Product life cycle cost analysis: state of the art review, International journal of production research, vol. 36, no. 4, pp. 883-908

[Ave98] Aven T., Jensen U., 1998, Stochastic models in reliability, Springer Verlag, New York

[Bav09] Bavaud F., 2009, Information theory, relative entropy and statistics, chapter in book: Formal Theories of information, Springer-Verlag, Berlin, pp 54-78

[Bel] Bellec J. H., Kechadi M.T, Carthy J., Performance Evaluation of Data Mining Techniques of Alarms Analysis, Internal Report, School of Computer Sciences & Informatics, University College Dublin, Belfield, Ireland

[Ben04] Bendixen M., Bukasa K. A., Abratt R., 2004, Brand equity in business to business market, Industrial marketing management, vol. 33, no.5, pp. 371-380

[Ben80] R. Benjamin, 1980, Some philosophical aspects of signal processing, Communications, Radar and Signal Processing, IEE Proceedings F ,Vol. 127, No. 2, pp. 67-75

[Bha03] Bharatendra R., Nanua S., 2003, Hazard rate estimation from incomplete unclean warranty data, Reliability Engineering and Systems Safety, vol. 81, pp. 79-92

[Bla06] Blanchard B.S., Fabrycky W.J., 2006, Systems Engineering and Analysis, New Jersey: Pearson Prentice Hall

[Bla98] Blanks H.S, 1998, The challenge of quantitative reliability, Quality and Reliability Engineering International, vol. 14, pp. 167-176

[Cao05] H., Mamoulis N., Cheung D.W., 2005, Mining frequent spatio-temporal sequential patterns, Fifth IEEE international conference on data mining, Houston, TX, pp. 82–89

[Cao07] Cao H., Mamoulis N., Cheung D.W., 2007, Discovery of periodic patterns in spatiotemporal, Knowledge and Data Engineering, IEEE Transactions on knowledge and data engineering

[Car98] J. Carreira, H. Madeira, J. G. Silva, 1998, Xception: A technique for the experimental evaluation of dependability in modern computers, IEEE Transactions on Software Engineering, vol. 24, No.2, pp. 125–136

[Che69] Cheetham A.H., Hazel J.E., 1969, Binary (presence-absence) similarity coefficients, Journal of paleontology, vol. 43, no. 5, pp. 1130-1136

[Che98] Chen X., Petrounias I., 1998, A framework for temporal data mining, Proc. Ninth International Conference on database and expert systems applications, DEXA '98, Vienna, Austria., Springer-Verlag, Berlin, Lecture Notes in computer science 1460, pp796-805

[Che69] Cheetham A.H., Hazel J.E., 1969, Binary (presence-absence) similarity coefficients, Journal of paleontology, Vol. 43, No. 5, pp. 1130-1136

[Cho04] Chopra S., Meindl P., 2004, Supply chain management, strategy, planning, and operations, Pearson Education Inc., New Jersey

[Cho07] Choi C.B., Song S., Koffler G., Medhi D., 2007, Outage analysis of university campus network, Proceedings of the 16th international conference on computer communication and networks 2007

[Cho90] Choi G. S., Iyer R.K., Carreno V.,A., 1990, Simulated fault injection: A methodology to evaluate fault tolerant microprocessor architectures., IEEE Transactions on reliability. vol.,39, no. 4, pp. 486-491

[Cin05] Cinque M., Cornevilli F., Cotroneo D., Russo S., 2005, An Automated Distributed Infrastructure for Collecting Bluetooth Field Failure Data, to appear in Proc. of the 8th IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC'05)

[Coh03] Cohen W.W., Ravikumar P., Fienberg S.E., 2003, A comparison of string distance metrics for name matching tasks, In Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03)

[Coi86] Coit D.W., Dey K.A., Turkowski W.E., 1986, Practical reliability data and analysis, Reliability Engineering, vol. 14, pp. 1-17

[Con99] Constantinescu C., 1999, Assessing error detection coverage by simulated fault injection, Lecture notes in computer science, Vol. 1667, pp. 161-170

[Coo79] Cooper R. G., 1979, The dimensions of industrial new product success and failure, The journal of marketing, vol. 43, no 13, pp. 93-103

[Cox62] Cox D. R., 1962, Renewal Theory, Methuen & Co. Ltd, UK

[Das97] Das G., Fleischer R., Gasieniec L., Gunopulos D., Karkkainen, J. 1997, Episode matching. In Proceedings of the 8th Symposium on Combinatorial Pattern Matching (CPM '97), Aarhus, Denmark, pp. 12–27

[Dav79] Davies D. L., Bouldin D.W., 1979, A cluster separation measure, IEEE transactions on patter analysis and machine intelligence. vol. PAMI-1, no. 2

[Djo95] Djoko S., Cook D. J., Holder L. B., 1999, Analyzing the benefits of domain knowledge in substructure discovery, In 1st ACM SIGKDD Intl. Conference on Knowledge Discovery and Data Mining, pp. 75-80

[Dua99] Duarte J.M., Santos J.B., Melo L.C., 1999, Comparison of similarity coefficients based on RAPD markers in the common bean, Genetic and molecular biology, vol. 22, no. 3, pp. 427-432

[Dub99] Dubois D., Prade H., 1999, Properties of measures of information in evidence and possibility theories, Fuzzy sets and systems, vol. 100, pp. 35-49

[End01] Endrenyi J. et al, 2001, The present status of maintenance strategies and the impact of maintenance on reliability, IEEE Transactions on power systems, vol. 16, no. 4, pp. 638-646

[Fay96] Fayad U., Piatesky-Shapiro G., Smyth P., 1996, Knowledge Discovery and data mining: towards a unifying framework, Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, Portland, OR, , pp. 82-88.

[Fer02] Ferring B. G., Plank A. E., 2002, Total cost of ownership models: an exploratory study, The journal of supply chain management, vol. 38, no. 3, pp. 18-19

[Fer03] Ferreira M. C. , Levkowitz H. , 2003, Form visual data exploration to visual data mining: a survey, IEEE Transactions on visualization and computer graphics, vol. 9. no. 3, pp. 378-394

[Fra92] Frawley W. J., Piatesky-Shapiro G., Matheus C. J., 1991, Knowledge discovery in databases: an overview, in G. Piatetsky-Shapiro and W. J.Frawley (eds.), Knowledge Discovery in Databases, AAAI/MIT Press, pp. 1-27

[Gai93] Gaines B.R., 1993, Modeling practical reasoning, International journal of intelligent systems, vol. 8, pp. 51-70

[Gow86] Gower J.C., 1986, Metric and Euclidean properties of dissimilarity coefficients, Journal of classification, vol.3, pp. 5-48

[Gre01] Greco S.,Masciari E.,Pontieri L., 2001, Combining inductive and deductive tools for data analysis, AI communications, vol. 14, pp. 69-82

[Gri67] Grizzle J.E., 1967, Continuity correction in the chi-square-test for 2x2 tables, The american statistician, vol. 21, pp. 28-32

[Gu03] Gu W., Kalbarczyck Z., Iyer R.K., Yang Z., 2003, Characterization of Linux Kernel Behavior under Errors, Proceedings of the 2003 International Conference on Dependable Systems and Networks, IEEE, 22 -25 June

[Ham03] Hamilton J., 2003, Active server availability feedback, Proceedings for the 2003 CIDR Conference

[Ham93] Hampton J.A., 1993, Prototype models of concept representation in In I. Van Mechelen, J. A. Hampton, R. S.Michalski, & P. Theuns (Eds.), Categories and concepts: Theoretical views and inductive data analysis, London: Academic Press. pp. 64-83

[Han92] Hansen J.P., Siewiorek D.P., 1992, Models for time coalescence in event logs, Proc. 22nd Int'l Symp. Fault-Tolerant Computing (FTCS-22), pp. 221-227

[Han00] Hand D.J., Blunt G., Kelly M.G., Adams N.M., 2000, Data mining for fun and profit (with discussion), Statistical Science, vol.15, pp. 111-131.

[Han88] Hansen J., 1988, Trend Analysis and Modeling of Uni/Multi-Processor Event Logs, Master's Thesis, Carnegie-Mellon University, Pittsburgh, PA

[Haw81] Hawkins D., 1981, An analysis of expert thinking, International journal of man machine studies, Vol. 18, pp. 1-47

[Ho94] Ho Y.C., 1994, Abduction? Deduction? Induction? Is there a logic of exploratory data analysis?, Annual meeting of American educational research association, New Orleans, Louisiana

[Hol06] E. Hollnagel, D.D. Woods, N. Leveson, 2006, Resilience Engineering Concept and Precepts, TJ International Ltd, Padstow, Cornwall, United Kingdom

[Hsu97] Hsueh, M.C., Tsai, T. K., Iyer, R. K., 1997, Fault injection techniques and tools. IEEE Computer, vol. 4, pp. 75-82

[Hub06] Hubert L., Arabie P., Meulman J., 2006, the structural representation of proximity matrices with Matlab, Society for industrial and applied mathematics, Philadelphia

[ISO01] ISO/IEC Standard 9126-1, Software Engineering – Product Quality – Part 1 : Quality Model, ISO Copyright Office, Geneva, June 2001

[Iye00] Iyer R. K., Kalbarczyk Z., Kalyanakrishnam M., 2000, Measurement-Based Analysis of Networked, System Availability. Performance Evaluation Origins and Directions, Ed. G. Haring, Ch. Lindemann, M. Reiser, Lecture Notes in Computer Science, 1769, Springer Verlag

[Iye82] Iyer R.K., Rosetti D., Hsueh M.C., 1986, Measurement and Modeling of Computer Reliability as Affected by System Activity, ACM Transactions on Computer Systems, vol. 4, no. 3, , pp. 214-213

[Iye86] Iyer R.K., Young L.T., Sridhar V., 1986, Recognition of error symptoms in large systems, Proceedings of 1986 ACM Fall joint computer conference, Dallas, Texas, United States, pp. 797-806

[Jai88] Jain A.K., Dubes R.C., 1988, Algorithms for clustering data, Prentice Hall Inc., New Jersey, USA

[Jay79] Jaynes E.T., 1979, Where do we stand on maximum entropy, in: Levine R.L., Tribus M., The maximum entropy formalism, MIT Press

[Jay82] Jaynes E.T., 1982, On the rationale of maximum-entropy methods, Proceedings of the IEEE, vol. 70, pp. 939-952

[Kal99] Kalyanakrishnam M., Kalbarczyk Z., Iyer R., 1999, Failure data analysis of a LAN of Windows NT based computers, In Proc.of the Symposium on Reliable Distributed Systems (SRDS'99),pp. 178–189, Washington - Brussels - Tokyo, Oct. IEEE.

[Kan92] Kanawati, G., Kanawati, N., Abraham, J., 1992, FERRARI: A Tool for the Validation of System Dependability Properties. Proc. 22nd FTCS Symposium, pp. 336-344

[Kee78] Keen P. G. W., Morton M. S. S., 1978, Decision support systems: an organizational perspective, Addison-Wesley publishing company

[Kei96] Keim D.A, Kriegel, P.,1996 Visualization techniques for mining large databases: A comparison, IEEE Transactions on knowledge and data engineering, vol. 8.,no6. pp. 923-936

[Kei01] Keim, D.A., 2001, Visual exploration of large data sets, Communications of the ACM, vol. 44., no. 8, pp. 38-44

[Kli87] Klir G.J., 1987, Where do we stand on measures of uncertainty, ambiguity, fuzziness, and the like?, Fuzzy sets and systems, vol. 24, pp. 141-160

[Lan69] Lancaster H.O., 1969, The Chi-squared distribution, John Wiley & Sons, New York

[Lap00] Lapierre J., 2000, Customer perceived value in industrial contexts, Journal of business & industrial marketing, vol. 15, no. 2/3, pp122-145

[Law92] Lawless J.F., Kalbfleisch J.D., Some issues in the collection and analysis of field reliability data, in: J.P.Klein, P.K. Goel (Eds.), 1992, Survival Analysis: State of the Art, Kluwer, Amsterdam, pp. 141–152.

[Laz92] Lazar A. A., Wang W., Deng R., 1992, Models and algorithms for network fault detection and identification: A review, In ICC

[Lev01] Levine E., Domany E., 2001, Resampling method for unsupervised estimation of cluster validity, Neural computation, vol. 13, pp. 2573-2593

[Lev66] Levensthein V.I., 1966, Binary codes capable of correcting deletions, insertions and reversals, Soviet phys. Dokl., vol. 10 , no. 8, pp. 707-710

[Lim08] Lim C., Singh N., Yajnik S., 2008, A log mining approach to failure analysis of enterprise telephony systems. In Proc. DSN

[Loh00] Loh S., Wives L. K., de Oliveira J. P. M., 2000, Concept-based knowledge discovery in texts extracted from the web, SIGKDD Explorations, vol. 2, no. 1, pp. 29-39

[Ma99] Ma S., Hellerstein J.L., 1999, Ordering categorical data to improve visualization, Proceedings IEEE symposium on information visualization

[Ma01] Ma S., Hellerstein J.L., 2001, Mining partially periodic event patterns with unknown periods, Proceedings 17th International Conference on Data Engineering

[Man97] Mannila H., Toivonen H., Verkamo A.I., Discovery of frequent episodes in event sequences, Data mining and knowledge discovery, vol. 1, no. 3, pp. 259-289

[Mar02] Mardsen E., Fabre JC. Arlat J., 2002, Dependability of CORBA systems: Service characterization by fault injection, 21st IEEE Symposium on reliable distributed systems, pp. 276-285

[Mar05] Markeset T., Kumar U., 2005, Product support strategy: conventional versus functional product, journal of quality in maintenance engineering, vol. 11, no. 1, pp. 53-67

[Mar93] Marzal A., Vidal E., 1993, Computation of normalized edit distance and applications, IEEE Transactions on pattern analysis and machine intelligence, vol. 15, no. 9, pp. 926-933

[Mat93] Matheus C.J., Chan P.K., Piatesky-Shapino G., 1993, Systems for knowledge discovery in databases, IEEE Transactions on knowledge and data engineering, vol. 5, no. 6, pp. 903-913

[Mil85] Milligan G.W., Cooper M.C., 1985, An examination of procedures for determining the number of clusters in a data set, Psychometrica, Vol 50, no. 2, pp. 159-179

[Mis06] Mishra K., Trivedi K. S., 2006, Model based approach for autonomic availability management. In Proc. Int. Symposium on Service Availability, ISAS, Helsinki, Finland

[Moj75] Mojena R., 1975, Hierarchical grouping methods and stopping rules: An evaluation, The Computer Journal, vol.20, pp. 359-363

[Mor90] Moran P., Saffney P., Melody J., Condon M., Hayden M., 1990, System Availability Monitoring," IEEE Trans. Reliability, vol. 39, no. 4, pp. 480-485

[Mud02] Mudambi S., 2002, Branding importance in business to business markets: There buyer clusters, Industrial marketing management, vol. 31, pp. 525-533

[Mur 95] Murphy B., Gent T., 1995, Measuring System and Software Reliability using an Automated Data Collection Process, Quality and Reliability Engineering International, vol. 11, pp. 341–353

[Oli03] Olivia R., Kallenberg R., 2003, Managing the transition from products to services, International journal of service industry management, Vol. 14, pp. 160-172

[Öne10] Öner K.B., 2010, Optimal reliability and upgrading decisions for capital goods, Proefschrift, University printing office, Eindhoven

[Pav04] Pavola S., 2004, Abduction as a logic and methodology of discovery: the importance of strategies, Foundation of Science, vol. 9, pp. 267-283

[Pat09] Patterson P.G., Spreng R. A., 1997, Modeling the relationship between perceived value, satisfaction and purchase intentions in a business-to-business, service context: an empirical examination, International journal of service industry management, vol. 8, no. 8, pp. 414-434

[Pei34/60] Peirce C.S., 1934/1960, Collected papers of Charles Sanders Peirce, Cambridge: Harvard University Press

[Pet03] Petkova V., 2003, An analysis of field feedback in consumer electronics industry, PhD thesis, Universiteitsdrukkerij Technische Universiteit Eindhoven

[Phi06] Philips Medical Systems Nederland B.V., 2006, Software Architecture Specification Rocket C, version 0.4

[Pro00] Prodromidis A., Chan P., Stolfo S., 2000, Meta-learning in distributed data mining systems: Issues and approaches. In H. Kargupta and P. Chan, editors, Advances in Distributed and Parallel Knowledge Discovery. AAAI/MIT Press, Cambridge, MA

[Qua00] Quach N., 2000, High Availability and Reliability in the Itanium Processor, IEEE Micro, vol.20, no.5, pp. 61-69.

[Reu10] Reuters, Sep 2010, Dutch medical isotope reactor to restart

[Rod02] Roddick J.F., Spiliopoulou M., 2002, A survey of temporal knowledge discovery paradigms and methods, IEEE transaction on knowledge discovery and data engineering, vol. 14, no. 4, pp. 750-767

[Rou87] Rousseeuw P.J., 1987, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, Journal of computational and applied mathematics, vol. 20, pp. 53-56

[Sha48] Shannon C.E.,1948, A mathematical theory of communication, Bell System Techn. Journal, vol. 27, pp. 379-423, 623-656,

[Sha76] Shafer Glenn, 1976, A mathematical theory of evidence, Princeton University Press, Princeton, New Jersey

[She94] Shen W.M., Mitbander B., Ong K., Zaniolo C., 1994, Using Metaqueries to Integrate Inductive Learning and Deductive Database Technology, In Proc. AAAI'94 Workshop Knowledge Discovery in Databases (KDD'94), pp. 335-346, Seattle, WA

[Sim05] Simache C., Kaâniche M., 2005, Availability assessment of sunOS/solaris unix systems based on syslog and wtmpx log files: A case study, In Pacific Rim Intl. Symp. on Dependable Computing, pages 49–56. IEEE Computer Society

[Sim96] Simoudis E., 1996, Integrating inductive and deductive reasoning for data mining, Advances in knowledge discovery and data mining, AAAI Press, London

[Tal99] Talagala N., Patterson D., 1999, An Analysis of error behaviour in a large storage system, In the workshop on fault tolerance in parallel and distributed systems

[Tel08] The Telegraph, February 2008, Heathrow engulfed by baggage chaos

[Tha96] Thakur A., Iyer R., 1996, Analyze-NOW--an environment for collection and analysis of failures in a network of workstations, IEEE Transactions on Reliability, R46(4)

[The06] Theodoridis S., Koutroumbas K., 2006, Patter Recognition Third edition, Academic Press, London, UK

[Tie00] Tierney B., Crowley B., Gunter D., Holding M., Lee J., Thompson M., 2000, A monitoring sensor management system for grid environments, In Proc. 9th IEEE Symp. on High Performance Distributed Computing, pp. 97–104

[Tri08] Trivedi K., Ciardo G., Dasarathy B., Grottke M., Rindos A., Matias R., Vashaw B., 2008, Achieving and Assuring High Availability, Proc. 13th IEEE Workshop on Dependable Parallel, Distributed and Network-Centric Systems/22nd IEEE International Parallel & Distributed Processing Symposium

[Tsa83] Tsao M.M., 1983, Trend Analysis and Fault Prediction, Ph.D. thesis, Carnegie-Mellon University, Pittsburgh, PA

[Van88] Vandermerwe S., Rada J.,1988, Servitization of Business: Adding value by adding services, European management journal, vol. 6, no. 4, pp. 314-324

[Vid88] Vidal E., Cascuberta F., Benedi J. M., Lloret M.J., Rulot H., 1988, On the verification of triangle inequality by dynamic time warping dissimilarity measures, Speech Communication, vol. 7, pp. 67-79

[Vro10] K. Vrotsou, 2010, Everyday mining, Exploring sequences in event based data, Dissertation No. 1331, LiU-Tryck, Linköping, Sweden

[Wag74] Wagner R.A., Fischer M.J., 1974, The string-to-string correction problem, Journal of the association for computing machinery, vol. 21, no. 1, pp. 168-173

[War08] Warrens M.J., 2008, On association coefficients for 2x2 tables and properties that do not depend on the marginal distributions, Psychometrica, Vol. 73, No. 4, pp. 777-789

[Wei90] Wein A., Sathaye A., 1990, Validating complex computer system availability models, IEEE Transactions on Reliability, vol. 39, no. 4, pp. 468–479

[Woo97] Woodward D. G., 1997, Life cycle costing-theory information acquisition and application, International Journal of project management, vol. 15, no. 6, pp. 335-344

[Yam05] Yamanishi K., Maruyama Y., 2005 Dynamic syslog mining for network failure monitoring, Proceedings 11th ACM SIGKDD, International Conference on Knowledge Discovery and Data Mining, ACM Press, New York, pp. 499-508

[Yan00] Yang J., Wang W., Yu P.S., 2000, Mining asynchronous periodic patterns in time series data, Proceedings of the 6th international conference on knowledge discovery and data mining

[Yuj07] Yujian L., Bo L., A normalized Levenshtein distance metric, IEEE transactions on pattern analysis and machine intelligence, vol. 29, no. 6, pp. 1091-1095

The product life cycle

The approach requires all phases of PLC to be taken into account when developing the product and to be incorporated into all processes from the very first identification of customer needs until the physical product is realized and is ready to be sold. This approach helps in reducing life cycle costs as well as lead time of the PSS readiness. This is achieved by looking at product requirements from a long term, life cycle perspective.

# Appendix A **The product life cycle**

To outline the approach the most basic phases of PLC need to be identified.

- PLC-Concept-Preliminary Design Phase (PDP)

During the initial stages of the PDP, the product concept is defined in respect to the customer needs in the form of requirements: operational requirements (when the product is in operation) e.g. functionality, capacity, compatibility, meta-operational requirements (for the product to be in operation) like reliability, maintainability, flexibility and other performance requirements as operating cost or environmental compliance. One of the objectives of this stage is to define the system specifications on top level and the product base-line, which will serve as the basis for further product development.

- Detailed Design and Development (PDP)

In a top-to-bottom approach technical requirements on system level are decomposed into all necessary hierarchical levels of the system's architecture.

- Production/Assembly

Material requirements have been established and suppliers have been identified. The production/assembly processes has to satisfy market demands with products that meet specifications.

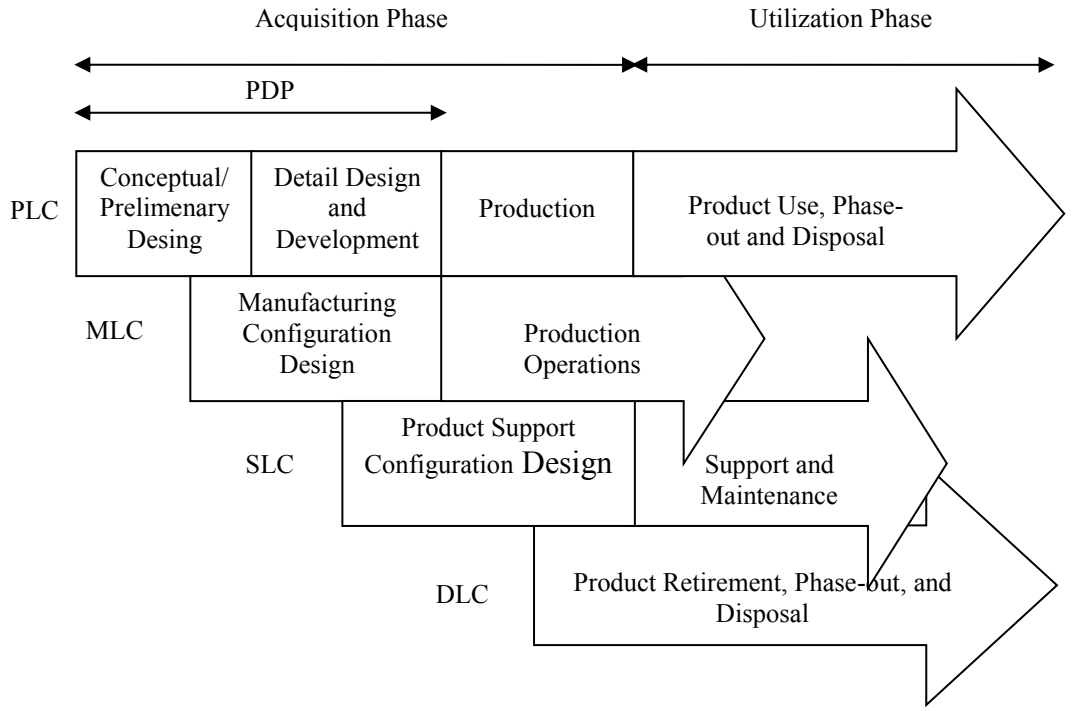



**Figure A-1: Product life cycle together with manufacturing, support and disposal life cycle**

- Utilization Phase

The product is in its economic useful life. The product needs to meet its TPMs and the PSS has to ensure that the product fulfills this requirement. However changes can be still introduced into the product's design either to improve the product's functionality or to correct design flaws if this is necessary.

- PLC – Phase Out and Disposal

  In this phase the product becomes obsolete and it is being replaced by newer versions. Product parts may be reused or will be disposed according to environmental regulations. Together with it the SLC comes to an end.

The first three phases form the *product development process* (PDP). The PDP together with the *production* phase is known as *product creation process* (PCP).

The complete PLC can be segmented into two phases, the *acquisition phase* where the OEM is responsible for all product related activities and the *utilization phase* where product management is shared between customer and OEM or third parties.

# Appendix B **The measure of merit $M_L$ for the resampling method [Lev01]**

To make the comparison of the membership of data points between the clustering results $C_p^R$ and $C_p^{R'}$, both results need to be represented in a suitable manner. The clustering result $C_p^R$ is represented with the help of a $N \times N$ cluster connectivity matrix $T_p$ defined by:

$$T_{p_{ij}} = \begin{cases} 1 & \text{points } i \text{ and } j \text{ belong to the same cluster} \\ 0 & \text{otherwise} \end{cases}$$

**( B-1)**

The connectivity matrix gives a very clear indication of the membership of data points in a cluster. Data points that belong to the same cluster are indexed with the value **one**, otherwise they a valued with **zero**. All elements on the diagonal are valued by definition with one (Table B-1). Assume the clustering result $C_p^R$ of the data sequence $S = \langle (t_1),(t_2),...,(t_9) \rangle$ for the value $\theta_p$ for the cutoff parameter. The clustering result is shown in Table B-1. It can be seen that the clustering result contains four clusters: $C_1 = (t_1, t_2)$, $C_2 = (t_3, t_4, t_5)$, $C_3 = (t_6, t_7)$, $C_4 = (t_8)$ and $C_5 = (t_9)$. The membership of the data points to the same cluster is indicated by the values of one in the upper diagonal of the matrix.

| | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $t_5$ | $t_6$ | $t_7$ | $t_8$ | $t_9$ |
|---|---|---|---|---|---|---|---|---|---|
| $t_1$ | **1** | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $t_2$ | | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $t_3$ | | | **1** | **1** | **1** | 0 | 0 | 0 | 0 |
| $t_4$ | | | | **1** | **1** | 0 | 0 | 0 | 0 |
| $t_5$ | | | | | **1** | 0 | 0 | 0 | 0 |
| $t_6$ | | | | | | **1** | **1** | 0 | 0 |
| $t_7$ | | | | | | | **1** | 0 | 0 |
| $t_8$ | | | | | | | | **1** | 0 |
| $t_9$ | | | | | | | | | **1** |

**Table B-1Example of connectivity matrix for original sequence $S$**

The original sequence $S$ is sampled randomly *k* times producing a set of secondary sequences $S' = S_1', S_2', ..., S_k'$ . For the resampling of $S$, a "dilution factor" *f* is used, that is the probability of sampling a data point in the sequence ( $0 \le f \le 1$). Because of the dilution the average distance between two points in $S'$ is scaled upwards by a factor *f* in relation to $S$. To compensate for that scaling, every subsequence $S'$ is

processed by the clustering algorithm but this time using adjusted cutoff value $\frac{\theta_p}{f}$. A set of cluster results $C_p^{R'} = \{C_{p1}^{'}, C_{p2}^{'}, ..., C_{pk}^{'}\}$ is obtained by clustering every $S_i^{'}$ in $S^{'}$.

Every clustering result in $C_p^{R'} = \{C_{p1}^{'}, C_{p2}^{'}, ..., C_{pk}^{'}\}$ is represented also by the corresponding connectivity matrix resulting to a set of connectivity matrices $T_p^{'} = \{T_{p1}^{'}, T_{p2}^{'}, ..., T_{pk}^{'}\}$, where $T_p^{'}$ is a $fN \times fN$ matrix.

In this example a dilution probability of $f = 0.5$ is used on $S$. Approximately half of the data points in the original data sequence are left out when resampled. In Table B-2 the clustering result of the resampling data sequence $S'$ is shown with its connectivity matrix $T_p^{'}$. For clarity the data points that are excluded in the resampling are marked with a red cross. Practically they do not exist in $S'$ anymore, but are included in the matrix to demonstrate the method. It can be seen that data points $t_2, t_4, t_8, t_9$ are excluded in $S'$. As a consequence the clusters now present in the clustering result are $C_1 = (t_1), C_2 = (t_3, t_5), C_3 = (t_6, t_7)$. The particular interest here lies with cluster $C_2$. In fact the data points $t_3$ and $t_4$ will end up in the same cluster only if $\theta_p$ is large enough to allow the algorithm to cluster these points together. Only then the data points that where in the same clusters in $C_p^R$ will remain in the same clusters also in $C_p^{R'}$ (One data point encircled in green). If $\theta_p$ is not large enough, the two data points will be assigned to different clusters and the original segmentation of the data sequence $S$ will not be matched. Data points that do not exist in the resampled data sequence do not interfere with the measure of merit.

| | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $t_5$ | $t_6$ | $t_7$ | $t_8$ | $t_9$ |
|---|---|---|---|---|---|---|---|---|---|
| $t_1$ | **1** | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $t_2$ | | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $t_3$ | | | **1** | **1** | **1** | 0 | 0 | 0 | 0 |
| $t_4$ | | | | **1** | **1** | 0 | 0 | 0 | 0 |
| $t_5$ | | | | | **1** | 0 | 0 | 0 | 0 |
| $t_6$ | | | | | | **1** | **1** | 0 | 0 |
| $t_7$ | | | | | | | **1** | 0 | 0 |
| $t_8$ | | | | | | | | **1** | 0 |
| $t_9$ | | | | | | | | | **1** |

**Table B-2Example of connectivity matrix for resampled sequence $S'$**

The measure of merit $M_{Lp}$ is computed by averaging the agreement between the connectivity matrix $T_p$ of the original date sequence $S$ and $T_p^{'}$ of the subsequences $S_p^{'}$ over all *k*.

The agreement between $T_p$ and $T_{pi}^{'}$ is expressed by

$$M_L(\theta_p) = \left\langle \left\langle \delta_{T_{ij}, T_{m_{ij}}^{'}} \right\rangle \right\rangle_k \qquad \textbf{( B-2)}$$

The number of data points that are members of the same cluster in $C_p^R$ and in $C_{pi}^{R'}$ is expressed by $\delta_{T_{ij}, T_{m_{ij}}^{'}}$ . This is averaged over all pairs of points *ij* that are members of the same cluster in the original clustering solution $C_p^R$ and survived the resampling. The operation is indicated by $\left\langle \delta_{T_{pij}, T_{pm_{ij}}^{'}} \right\rangle$ . Another averaging operation is performed over all $k$ and $M_{Lp}$ is derived for a given value of $\theta_p$ . $M_L$ is taking values $0 \le M_L \le 1$, with $M_L = 1$ being the perfect score for total agreement.

# Appendix C **Definition of matrices $P$, $\bar{T}_C$ and the test statistic $\hat{\Gamma}$**

- The proximity matrix $P$ is a measure of the internal structure of the data sequence $S$, and is given by:

  $$P_{ij} = d(x_i, x_j) \qquad \textbf{( C-1)}$$

  , where $d(x_i, x_j) = |x_i - x_j|$, the Euclidean distance between points $x_i$ and $x_j$.

- The dis-connectivity matrix for the clustering result $C, \bar{T}_C$ is defined as follows:

  $$\bar{T}_{C_{ij}} = \begin{cases} 1 & \text{points } i \text{ and } j \text{ belong to different clusters} \\ 0 & \text{otherwise} \end{cases} \qquad \textbf{( C-2)}$$

  for $i, j = 1, 2, ..., N$

- The correlation between the $P$ and $\bar{T}_C$, is given by:

  $$\hat{\Gamma} = \frac{(1/M)\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}(P(i,j) - \mu_P)(\bar{T}_C(i,j) - \mu_{\bar{T}_C})}{\sigma_P \sigma_{\bar{T}_C}} \qquad \textbf{( C-3)}$$

  where $M = N(N-1)/2$ is the number of pairwise elements in $P$ or $\bar{T}_C$ (the statistic is using the upper diagonal elements of the matrices).

  The mean value $\mu_P$ for $P$ is given by:

  $$\mu_P = (1/M)\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} P(i,j) \qquad \textbf{( C-4)}$$

  and variance $\sigma_P^2$:

  $$\sigma_P^2 = (1/M)\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} P(i,j)^2 - \mu_P^2 \qquad \textbf{( C-5)}$$

  Correspondingly for $\bar{T}_C$ the mean $\mu_{\bar{T}_C}$

  $$\mu_{\bar{T}_C} = (1/M)\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} \bar{T}_C(i,j) = \frac{m}{M} \qquad \textbf{( C-6)}$$

  where $m$ the number of pairs of points that belong to different clusters and variance $\sigma_{\bar{T}_C}^2$:

  $$\sigma_{\bar{T}_C}^2 = (1/M)\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} \bar{T}_C(i,j)^2 - (\frac{m}{M})^2 = \frac{m(M-m)}{M^2} \qquad \textbf{( C-7)}$$

**The $\hat{\Gamma}$ statistic takes values $0 \le \hat{\Gamma} \le 1$, with $\hat{\Gamma} = 1$ being the perfect match between matrices *P* and $\bar{T}_C$.**

$\hat{\Gamma}_{r_i}$ is calculated for every $C_{r_i}$ in the same way producing the set.

# Appendix D **Agglomerative clustering algorithm**

The input is the collection of tags $LC_1 = \bar{C}^R$ . At this point, each tag is considered to form one cluster. At the first iteration of the algorithm two clusters $C_a$ and $C_b$, $a,b = \{1,2,3,...,M\}, a \neq b$, are merged into one new cluster $C_{M+1}$. This results to a new clustering result $LC_1$ with $M-1$ number of clusters. The process continues until the result $LC_{M-1}$ is obtained where one single cluster $C_{2M+1}$ contains all tags.

Another input to the clustering algorithm is the distance matrix $H_1 = H(LC_1)$. $H_1$ is a $M \times M$ matrix containing the pair wise distances between of tags in $LC_1$. At each iteration, when two clusters are merged into one the distance matrix is updated. The matrix then decreases in size by one unit in both dimensions by deleting the rows and columns of the clusters $C_a$ and $C_b$ that where merged and by inserting a new row and one column for the newly created cluster $C_q$. The distance between the newly created cluster $C_q$ and any other cluster $C_s$ in $LC_q$ is given by:

$$d(C_q, C_s) = \max(d(C_a, C_s), d(C_b, C_s)) \quad \textbf{( 0-3)}$$

The pseudo code for the agglomerative complete link algorithm is shown below

- $q = 1$
- $LC_1 = \bar{C}^R$
- $H_1 = H(LC_1)$
    - for $q = 1 to M - 1$
        - Find $C_i$ and $C_j$ such that $d(C_i, C_j) = \min_{i \neq j}(LC_q)$
        - Merge $C_i$ and $C_j$ into $C_{q+1}$ and form $LC_{q+1} = (LC_q - \{C_i, C_j\}) \cup \{C_q\}$
        - Produce $H_{q+1}$ using the function $d(C_q, C_s) = \max(d(C_i, C_s), d(C_j, C_s))$

## Curriculum Vitae

Kostas Kevrekidis was born on the 9th of February1976 in Kavala, Greece. After receiving in 2002 his Bachelor's degree in Automation Engineering at the Technical Institute of Thessaloniki in Greece, he graduated in 2005 with a Master's degree in Quality Management from the School of Science of the University of the West of Scotland in Paisley, Scotland. In 2005 he started a PhD project at the Eindhoven University of Technology, of which the results are presented in this dissertation. Since 2010 he is employed as a senior reliability engineer at TomTom International BV.