# User Behavior Pattern Based Security Provisioning for Distributed Systems

by

Weina Ma

A thesis

presented to

University of Ontario Institute of Technology

in fulfilment for the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Oshawa, Ontario, Canada, 2016

## Abstract

Behaviors of authorized users must be monitored and controlled due to the rise of insider threats. Security analysts in large distributed systems are overwhelmed by the number of system users, the complexity and changing nature of user activities. Identifying user behavior patterns by analyzing audit logs is challenging. Lacking a general user behavior pattern model restricts the effective usage of data mining techniques. Limited access to real world audit logs due to privacy concerns also blocks user behavior leaning. The central problem addressed in this thesis is the need to assist security analysts obtain deep insight into user behavior patterns.

To address the research problem, the thesis defines a user behavior pattern as consisting of four factors: actor, action sequence, context, and time interval. Based on this behavior pattern model, the thesis proposes a knowledge-driven user behavior pattern discovery approach, with step-by-step guidance for security analysts throughout the whole process. The user behavior pattern mining process are all uniformly represented using a formalism. A user/tool collaborative environment on top of data mining techniques is designed for constructing a baseline of common behavior patterns to individuals, peer groups, and specific contexts.

A prototype toolkit that is developed as part of this thesis provides an environment for user behavior pattern mining and analysis. To evaluate the proposed approach, a behavior-based dataset generator is developed to simulate audit logs containing designed user behavior patterns. Moreover, two real world datasets collected from distributed medical imaging systems and public cloud services are respectively applied to test the proposed model.

**Keywords:** User behavior pattern, data mining, synthetic dataset generation, security provisioning

# Author's Statement

I, Weina Ma, hereby declare that I am the sole author of this thesis.

# Acknowledgements

I would like to express my sincere appreciation to Dr. Kamran Sartipi, my supervisor for the first two and half years, for giving me the opportunity to pursue my PhD degree, providing fund and guidance throughout the entire process. Dr. Sartipi spent countless hours discussing different aspects of my research and helped to complete this research. Without Dr. Sartipi's guidance and constant feedback this thesis would not have been completed. Also, special gratitude to Dr. Ramiro Liscano, my supervisor at last year, for his helpful recommendation and continuous support and made this PhD achievable. I also would like to thank my committee members, Dr. Jeremy Bradbury and Dr. Masoud Makrehchi for agreeing to serve on my thesis and for their valuable expertise and precious time. I gracefully acknowledge the funding received from ORF (Ontario Research Fund) project, Secure Intelligent Content Delivery System for Timely Delivery of Large Data Sets in a Regional/National Electronic Health Record, that was initiated from McMaster university. Finally, thank you to the University of Ontario Institute of Technology (UOIT) for allowing me to conduct my research and providing remarkable support.

# Glossary

**insider threat** is a malicious threat to an organization that comes from people within the organization

**UBA** User Behavior Analytics

**user behavior pattern** refers to consistent observations of a sequence of actions that a user or peer groups conducted under certain context within a time interval

**common behavior pattern** is frequently occurring user behavior patterns

**frequent itemset** is a group of items occurring frequently in event database

**frequent itemset mining** is a algorithm for discovering frequent itemsets in event database

**MAG** (Maximum Association Group) is the maximum set of events that all share the same itemset

**association of MAG** measures the closeness of events in the same MAG

**association between MAGs** measures the closeness of events in one MAG to the events in another MAG

**BPLQ** (Behavior Pattern Query Language) describes the high level of event clusters with intra-cluster and inter-cluster constraints

**frequent sequential pattern** is a group of subsequences occurring frequently in sequence database

**frequent sequential pattern mining** is a algorithm for discovering frequent sequential patterns in sequence database

**LCS** (Longest Common Subsequence ) is a algorithm of finding the longest subsequence common to all sequences in a set of sequences

**sequence clustering** is an algorithm of collecting sequences in groups such that sequences in the same group are more similar to each other than to those in other groups

**representative sequence** is a set of non-redundant typical sequences that largely cover the sequences in a cluster

**dataset generator** is a simulation tool to generate events with embedded user behavior patterns, which is developed as part of the thesis

**Z notation** is is a formal specification language used for describing and modelling computing systems

**PACS** (Picture Archiving and Communication System) is medical imaging systems including acquiring, transmission, and storing images and diagnostic report

**AWS CloudTrail** is a web service that records Amazon AWS API calls and delivers log

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Despite almost $80 billion spent globally on security in 2015 [1], attackers are still penetrating various organizational defense. Particularly, insider threat is rising dramatically mainly because the impressive profit of leaking sensitive information, the increased use of cloud based applications that may leak data (e.g., web email, dropbox, social media), and the exploding data growth that is leaving the protected boundary [2].

With the availability of common security mechanisms such as authentication, authorization and secure communication in most systems, authorized users still intentionally or carelessly demonstrate risky behaviors that may cause data leakage or damage to the protected resources. The risk of insider threat to data security is growing, and nearly two-thirds of attacks or data leakage are originated from insiders recently [3, 4]. Security consciousness is shifting from traditional perimeter defence to holistic behavioral solutions to detect ongoing insider threats, and even prevent them before they actually happen [5, 6, 7, 8].

Since insiders already possess the privileges to access to the organization's information and assets, it is even harder to defend against than attacks from outsiders. Therefore, activities of authorized users must be constantly monitored and controlled. It has been a big challenge for system administrators in large distributed systems to identify user's behavior pattern, due to: the large number of users in using the system services including employees, contractors, customers, and partners; the complexity and changing nature of user behavioral patterns; and the lack of fully featured signature-based detection capabilities.

Understanding normal behavior is the key to determining insider threats. Ideally, the observed user behavior should be similar to his past behavior or peer group's behavior. The significant deviation indicates possible malicious threat. This thesis contributes to common user behavior pattern mining using unsupervised machine learning techniques, which can be effectively applied to unlabelled data in different application domains. The output of this thesis research, extracted common behavior patterns, might be used for insider detection by comparing with user's run-time activities.

## 1.1 User Behavior Analytics

"User Behavior Analytics as defined by Gartner, is about detection of insider threats, targeted attacks, and financial fraud. UBA solutions look at patterns of human behavior, and then apply algorithms and statistical analysis to detect meaningful anomalies from those patterns - anomalies that indicate potential threats" [9].

UBA collects various types of data such as organization structure, user roles and job responsibilities, privacy legislation, user activity trace and geographical location. The analysis algorithms consider factors including contextual information, continuous activities, duration of sessions, and peer group activity to compare anomaly behavior [10]. UBA determines the baseline of normal behavior of individual user or peer group according to history data. The deviation of ongoing user activities compared with past normal behavior is significant if the user acts abnormally.

Following are typical research problems have to be addressed in UBA: select and extract behavioral elements hidden in various data; develop normal behavior modeling and representation methods to capture behavior characteristics; propose effective techniques and tools for user behavior analysis; determine what are normal behavior, and what constitutes behavior anomalies; evaluate potential impact and risk of behavior anomalies; simulate user behaviors to understand all of the above mechanisms.

In this thesis we cast normal behavior modeling and analysis as contextual sequential pattern mining problem, and pose an interactive approach for efficiently guiding system administrators in identifying meaningful user behavior patterns.

## 1.2 Motivations

The number and variety of services in organizations are constantly increasing, and different distributed systems tend to integrate their services to form federated services; therefore, understanding user behavior is just becoming more important and complex. The challenge for system administrators has always been not knowing what to look for, how to represent user behavior patterns, and how to discover meaningful user behavior patterns from large dataset efficiently. The motivation for this research stems from following concerns:

- Applying a developed user behavior learning system to another application domain is not straightforward. A generic solution that is applicable to different application domains is required.

- The lack of reflective and uniform model for user behavior analysis, whereby the user behavior pattern, and pattern mining process are all uniformly represented using a formalism.

- The lack of automatic tools for user behavior pattern mining from large-scale unlabeled data.

- The lack of knowledge-driven approaches and processes for guiding system administrators step by step to explore user behavior patterns.
- The general lack of real-world high quality dataset. Developing a testing benchmark, user controllable toolkit to simulate dataset containing embedded interesting patterns and features, is quite useful to evaluate user behavior pattern mining approaches.

## 1.3 Thesis Statement and Challenges

One traditional behavior learning approach is using supervised machine learning techniques to build a prediction model, which heavily rely on high quality labeled training data. Unfortunately, labeled data is always short in supply in real-world applications. Also the trained model in one application domain usually cannot be reused in another application domain. An alternative approach is unsupervised machine learning that can be efficiently applied to unlabeled data. However, there is no goal to achieve in unsupervised learning process so that the outcome is unknown. Domain knowledge has been successful in improving unsupervised learning. Expressing and incorporating the domain knowledge into the user behavior recovery process is the first challenge of the research. Without prior knowledge about the dataset, unsupervised learning approach may produce a large number of irrelevance patterns which makes the result very difficult to be analyzed. The second challenge is developing a knowledge-driven process and environment for guiding system administrators step-by-step in establishing meaningful user behavior patterns from unlabeled data. Another major challenge is to perform evaluation on unsupervised user behavior pattern learning approach.

The thesis statement is defined as:

> *The common behavior is defined as frequent patterns that are discovered from the dataset. We cast user common behavior learning as a contextual frequent sequential pattern mining problem. We propose a behavior model, as well as an efficient common behavior extraction method without exhaustive mining. The proposed mining method improves the accuracy, recall and efficiency compared with normal frequent mining technique. By extracting a small set of categorized representative patterns, the result of this thesis work provides valuable knowledge to anomaly detection research.*

## 1.4 Proposed Solution

In this thesis, we consider a hypotheses: frequently occurring user behaviors that are discovered from a large event dataset are regarded as user common behaviors. If a specific behavior is repeatedly performed by user himself, or by peer group, most probably it is a common behavior. The discovered common behavior patterns of either an individual user or a group of users based on audit history are valuable knowledge to system administrators in designing insider detection approach.

### 1.4.1 Abstract Behavior Pattern

Accurate and comprehensive user behavior pattern abstraction is required first. We propose a generic behavior model that allows system administrator to map domain specific access logs onto attributed events and consequently a behavior model. We define a behavior as: *consistent observations of a sequence of actions that an actor conducted in a common context during a specific time interval* (e.g., a session, a day, a week). The user behavior pattern is represented using sequencing, timing and association rules: sequencing requires that a series of steps occur in a certain order; timing limits the occurrence frequency of user behavior; and association enforces that the occurrence of one value should result in one or more other values which constitute the behavior context.

### 1.4.2 Behavior Pattern Mining

Sequential pattern mining is a data mining technique that is used to discover the time-based correlations among customer transaction history. In this thesis, an ordered list of events generated by the same user constitutes a sequence. The aim of discovering frequent sequential patterns among user's event sequences is to uncover sequences of user actions that occur frequently. The task of discovering frequent sequential patterns from audit logs of large system is challenging, because the algorithm needs to process an explosive number of possible subsequences.

To reduce the search space for sequential pattern mining without significant information loss, we applied association mining operation as pre-processing on event database to discover highly related event groups. Based on the association mining result, we propose an association-based similarity metric to measure the similarity between events in the same associated group and between groups. We also designed a behavior pattern query language capable of expressing analyst's interest and focus. Using an iterative process, the analyst generates BPQ (Behavior Pattern Query) to collect highly related events into constrained clusters according to the suggested attributes, association group of events, and user's domain knowledge. The cluster constraints are regarded as the common context of all frequent action-sequences extracted from this cluster. The proposed approach is capable of discovering more valuable user behavior patterns with focus from limited search space.

User behavior studying focuses on modeling normal behavior patterns and identifying meaningful anomalies. Figure 1.1 presents this thesis work, and how

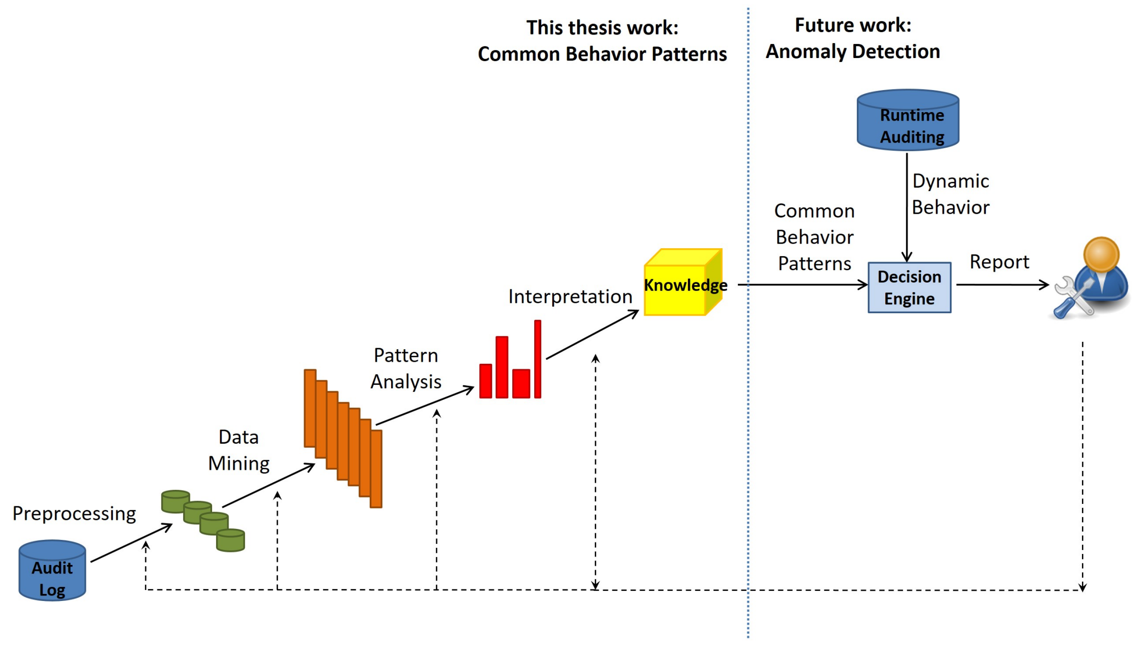


Figure 1.1: User common behavior pattern mining and anomaly detection: an iterative and interactive process

to apply the thesis work output into anomaly detection.

The left part of Figure 1.1 explains common user behavior pattern mining process of this thesis work, which applies data mining, pattern analysis, and knowledge representation approaches to build user common behavior patterns. The audit logs are first processed into attributed events. Data mining operations are then applied to the pre-processed database to compute the frequent sequential patterns under certain context. The context is semi-automatically constructed by data mining algorithm together with user specified constraints. One of the common criticisms pointed out to frequent pattern mining is the fact that the algorithm generates a huge number of patterns, making it very hard to analyze and use the result. Clustering and extracting representative techniques are then used to inductively learn the most representative user behavior patterns under certain context. The discovered knowledge is represented using our proposed abstract behavior model which is discussed in subsection 1.4.1. Several iterations of the data mining and analysis process may be required to obtain reasonable contextual common behavior patterns.

The right part of Figure 1.1 shows how the discovered common user behavior patterns fit into the bigger picture of anomaly detection in future work. User's dynamic run-time activities are monitored and sent to decision engine periodically. Decision engine compares user's dynamic behavior with corresponding common behavior patterns to decide whether or not to report the current dynamic behavior as an anomaly to the system administrators. The decision engine report provides valuable actionable insight. System administrators may investigate the properties of the recovered common behavior patterns and false anomaly report, and feedback to any step of common behavior pattern mining process to optimize the normal behavior baseline.

### 1.4.3 Testing Benchmark

There is a general lack of access to real-world audit logs, and in particular in sensitive and mission-critical industries, such as healthcare, banking, and military. Moreover, the performances of different data mining approaches depend heavily on the testing dataset. Some algorithms are very sensitive to the target patterns and critical features. A controllable simulation environment which enables acquiring various simulated dataset with embedded interesting patterns and features is required. We developed a dataset generator toolkit *EventGenerator* for controllable dataset generation, suitable for unbiased evaluation of user behavior pattern mining algorithms. *EventGenerator* has three layers: i) behavior pattern representation layer; ii) dataset generation layer; and iii) dataset visualization and analysis layer. We use *EventGenerator* as a testing benchmark for evaluation through generating synthetic dataset with designed pattens embedded, and then applying our proposed approach to recover the embedded behavior patterns.

## 1.5 Thesis Contributions

This thesis presents an approach for user behavior pattern mining that employs techniques from data mining, clustering, data visualization, pattern analysis, and pattern representation. The major contributions of this thesis are as follows:

1. A user behavior pattern discovery environment that can be applied on large distributed systems. Without any labeled training data and prior knowledge, the proposed system guides system administrators to obtain deep insight into the behavior patterns of the system users for security purposes.
    - Proposed a new behavior model to represent user behavior pattern as a combination of sequencing, association and timing rules.
    - The user behavior pattern mining process are all uniformly represented using a formalism.
    - Designed a behavior pattern query language (BPQL) that allows system administrators to describe a contextual behavior pattern to be discovered by the pattern mining engine.
2. A developed behavior-based dataset generator toolkit *EventGenerator* that allows data analysts to easily design and generate different testing datasets with embedded user behavior patterns. This toolkit provides a testing benchmark to evaluate behavior pattern mining approaches.

## 1.6 Thesis Structure

The remaining chapters of this thesis are organized as follows:

- **Chapter 2** provides an overview of different background knowledge and technologies that are employed in this thesis.

- **Chapter 3** starts with a dry run example to explain the maximal association relationship and proposed behavior pattern mining method. Chapter 3 also discusses the abstract behavior pattern model, a new behavior pattern query language to abstract high-level behavior pattern, and the proposed methodology in using data mining techniques to identify user common behavior patterns.
- **Chapter 4** formalizes common behavior pattern mining proeess using Z notation [11, 12]. Chapter 4 also introduces two association-based similarity measures at data instance level and data group level for event clustering.
- **Chapter 5** presents a synthetic dataset generator that effectively assists data analysts in designing behavior-based datasets with embedded user behavior patterns, and visually analyzing the generated datasets. A prototype toolkit *EventGenerator* has been developed to synthesize and analyze the datasets in different application domains.
- **Chapter 6** presents the experimentation with two real-world dataset from medical imaging system and public cloud services respectively, and a synthetic event dataset with embedded user behavior patterns. The experiments are divided into three case studies to demonstrate the functionality, performance and accuracy of the proposed user behavior pattern recovery technique.
- **Chapter 7** provides a conclusion for the whole thesis and outlines some potential future directions of this research.

# Chapter 2

# Background

In the chapter we present an overview of different background knowledge required to this thesis in user behavior pattern based security provisioning.

## 2.1 Anomaly Detection

Anomaly detection has been widely researched in various application domains. Intrusion detection is the process of monitoring network or system activities for identifying as well as responding to malicious activities [13]. Beside of host based and network based intrution detection systems, anomyly detection techniques are also used to detecte attacks against web servers and web-based applications. Fraud detection in business transactions, finacial and insurance industry is another application domain that anomaly detection is applied to identify fraud as it happens [14]. In healthcare system, a behavior-based access control model is proposed, which captures the dynamic behavior of the user, and determines access rights through comparing with the expected behavior [15].

"Anomaly detection refers to the problem of finding patterns in data that do not conform to expected behavior. The non-conforming patterns are often referred to as anomalies or outliers in different application domains." [16] Four key challenges with anomaly detection are: i) defining a whole set which includes all possible normal behavior is infeasible; ii) producing clear and precise boundary between normal behavior and anomaly is difficult; iii) acquiring labeled data for training and evaluating anomaly detection approaches is expensive; and iv) applying a technique developed in one domain to another is not straightforward because of the dynamic nature of malicious behavior [16].

Supervised and unsupervised machine learning algorithms are widely applied in anomaly detection approaches[17]. The input of anomaly detection system is a collection of data instances that can be described using a set of domain specific attributes. For supervised anomaly detection, the training data instances have already been associated with labels, denoting if that instance is normal or not. Typical approach in such cases is to build a predictive model using machine learning algorithm that learns specific properties from training data including both normal properties and anomaly properties. Unknown testing data instance is compared with the predictive model to identify which category it belongs to. Unsupervised anomaly detection techniques do not require any training data, but have an implicit assumption that normal instances are the majority in unlabelled data. Unsupervised machine learning algorithm learns from unlabelled data to identify hidden patterns and properties, which heavily depends on instance distance, density and statistics. If the unlabelled data has more anomalies than normal instances, the prediction model suffers from high false alarm rate. As a result of anomaly detection approach, each testing data instance is labelled as normal or anomalous; or each testing data instance is assigned an anomaly score which provides knowledge for the next analysis step.

## 2.2 Frequent Pattern Mining

Frequent patterns include itemsets, subsequences and substructures that appear in a dataset with frequency no less than a user specified threshold [18]. For example, a set of purchase items, such as milk and bread, which appear frequently together in a transaction dataset, is a frequent itemset. A subsequence, such as buying first a TV, then a DVD player, and then various CDs and DVDs, if it occurs frequently in a shopping history database, is a frequent sequential pattern. A substructure refers to different structural forms, such as subtrees, subgraphs, or sublattices, which may be combined with itemsets or subsequences. If a substructure occurs frequently in a graph database, it is called a frequent structural pattern. Frequent pattern mining plays an essential role in mining associations, sequences, and many other interesting relationships among data.

### 2.2.1 Association Mining

The concept of association mining was first introduced by Agrawal [19] in the form of association rules mining, aiming at analyzing customer purchase habit by extracting associations between items in customer shopping baskets. An itemset is a set of items that frequently appear in shopping baskets. An itemset with the cardinality of $k$ is called *k-itemset*. The support of an itemset is the number of transactions that contain that itemset in the transaction database. The Apriori algorithm [19] passes over the transaction database multiple times to discover frequent *k-itemsets* that appear in transactions more than a user specified threshold, namely minimum support (*minsup*). In the first pass, the algorithm counts the support of individual items and determines the frequent *1-itemsets*. In each subsequent pass, the algorithm selects different frequent *(k-1)-itemsets* found in the previous pass to generate candidate *k-itemsets* by joining those frequent *(k-1)-itemsets*. The candidate *k-itemset* will be deleted from candidate list if any of its subsets is not frequent, i.e., each subset of a candidate itemset must itself be frequent. Each candidate *k-itemset* must also have minimum support in order to be considered as "frequent k-itemset". This iterative process will terminate when the algorithm cannot generate any larger frequent *k-itemset*.

### 2.2.2 Sequential Pattern Mining

Agrawal [20] introduced sequential pattern mining of frequently occurring ordered events or subsequences. Consider a database of customer transactions where each transaction contains a customer-id, transaction time, and the items bought in the transaction. The collection of a customer's transactions, where each transaction contains a set of items and the transactions are ordered by increasing transaction time, are together viewed as a *customer-sequence*. Mining sequential patterns is the process of finding the frequent subsequences that appear in the customer-sequences more than a user specified threshold namely minimum support (*minsup*).

In association mining, the support for an itemset is defined as the number of transactions in which an itemset is present, whereas in the sequential pattern mining the support for a sequence is the number of customer-sequences that contain the sequence as a sub-sequence. The sequential pattern mining algorithm *Apriori* [20] passes over the sequence database multiple times. In the first pass, the algorithm finds the frequent *1-length* sequences. In each subsequent pass, the algorithm generates the candidate *k-length* sequences by joining frequent *(k-1)-length* sequences found in the previous pass, and then measures their support to determine whether they are frequent *k-length* sequence or not. A frequent *k-length* sequence must have minimum support (i.e., customer-sequences that contain the sequence as their subsequence). The process continues until the algorithm cannot find any larger frequent *k-length* sequence from the existing sequences. Having found the set of all frequent sequences, the algorithm removes the frequent sequences that are contained within the larger frequent sequences, as they are redundant.

Frequent pattern mining explores strong association among instances have been successfully applied in various application domains: user navigational behavior mining via analyzing Web log data [21]; software failure pattern detection [22]; and behavior-based access control systems [15]. A large number of recent studies have contributed to extending association mining and sequential pattern mining, including: constraint-based association mining and sequential pattern mining [23], multi-dimensional sequential pattern mining [24], context-based sequential pattern mining [25], and frequent episode discovery in event sequences [26].

## 2.3 Frequent Pattern Based Anomaly Detection

Frequent pattern mining based anomaly detection methods propose to mine all frequent patterns from temporal data in order to compute the anomaly factor for incoming behavior [27, 28, 29, 30]. A two-step procedure is typically involved: i) model the normal behavior across time using frequent pattern mining technique; ii) evaluate the anomaly degree of incoming behavior by measure the deviation from the discovered normal behavior. However, the goal of discovering all frequent patterns is too expensive, especially for mining temporal data. Despite all frequent patterns can be efficiently discovered, it is unfeasible to compare incoming behavior with all discovered normal patterns to detect anomalies instantly at the second step.

To address these issues, we applied i) association mining prior to sequential pattern mining to reduce search space while making sure the information loss is as small as possible; ii) clustering operation to categorize the normal behavior patterns so that anomaly detection step only selects related normal behaviors to measure deviation rather than all of them; and iii) representative sequences mining to reduce the number of discovered normal behavior patterns but still this small set of patterns satisfy a desired coverage.

## 2.4 Clustering

Data clustering is a method of grouping objects in a way that objects in one cluster are very similar to each other, but they are dissimilar to the objects in other clusters [31]. Clustering is unsupervised learning approach that partitions data into a certain number of clusters with little or no prior knowledge. The resulting clusters are the matter of interest. Two typical cluster models are: centroid models such as k-means algorithm in which a data instance belongs to the cluster with the nearest mean [32]; density models which define clusters as contiguous regions of high density in the data space [33]. Both centroid models and density models require a similarity or distance measure used for clustering. Similarity or distance measure describes quantitatively how close two data instances are.

### 2.4.1 Constrained Clustering

Unsupervised clustering is a subjective process because of subjectively chosen of similarity measure [34]. Semi-supervised learning, learning from a combination of unlabeled data and a small amount of labeled data, produces considerable improvement in learning accuracy [35]. For example, constrained k-means clustering shows significant improvement in accuracy, which incorporates background knowledge in the form of instance-level constraints into the k-means algorithms [36].

Constrained clustering is a type of semi-supervised learning technique that injects background knowledge (e.g., analyst's experience or knowledge of testing dataset) into clustering algorithm. Instance-level constraints are used to express a prior knowledge of which instances should be grouped together or be separated. For example, "*must-link*" constraint specify that two instances have to be in the same cluster; and "*cannot-link*" constraint specify that two instances have to be in different clusters [36]. Consequently, the constrained clustering algorithm takes in a dataset, a set of "*must-link*" and "*cannot-link*" constraints, a similarity measure between instances, a density threshold and the number of expected clusters; and returns a partition of the dataset that the instances in the same cluster are more similar to each other, at the same time they satisfy all specified constraints. Some constrained clustering algorithms will abort if it cannot partition the data without any constraint violation. Soft constrained clustering algorithms are applied when hard constraind clustering algorithm brings failure, which allows constraint violation with minimal cost [37].

### 2.4.2 Sequence Clustering

A survey of time series data clustering introduces general-purpose clustering algorithms commonly used in time series clustering studies, and the measures to determine the similarity/distance between two time series point [38]. We applied one of the introduced clustering algorithm into our approach: agglomerative hierarchical clustering. This clustering algorithm starts by placing each sequence as an individual cluster and then merges these clusters into larger clusters, until all sequences are merged into a single cluster or until other termination conditions are satisfied. The algorithm measures the similarity between two clusters as the closet (or farthest, average) pair of sequences belonging to separate clusters, and merging the clusters having the minimum distance. One of the advantages of agglomerative hierarchical clustering algorithm is no apriori knowledge about the desired number of clusters is not required.

The similarity problem of sequential data requires determining whether different sequences have similar behavior. An obvious measure of the closeness of two sequences is to find the maximum number of identical items in these two sequences (preserving the symbol order), which is defined as Longest Common Subsequence (LCS) of the sequences [39]. Formally, let $X = (x_1, x_2, ..., x_m)$ and $Y = (y_1, y_2, ..., y_n)$ be two sequences of lengths $m$ and $n$, respectively. Common subsequence "cs" of $X$ and $Y$ represented by $cs(X, Y)$ is a set of subsequences that occur in both sequences. The longest common subsequence $lcs$ of sequence $X$ and $Y$, $lcs(X, Y)$ is a common subsequence of both sequences with maximum length. The length of $lcs(X, Y)$ is denoted by $r(m, n)$. Solving $r(m, n)$ is to determine the longest common subsequence for all possible prefix combinations of the two sequences $X$ and $Y$. Let $r(i, j)$ be the length of the longest subsequence $lcs(X_i, Y_j)$, where the prefixes of $X$ are $X_i = (x_1, x_2, ..., x_i)$ and the prefixes of $Y$ are $Y_j = (y_1, y_2, ..., y_j)$. Then $r(m, n)$ can be defined recursively as following [39]:

$$r(i,j) = \begin{cases} 0 & \text{if } i = 0 \vee j = 0 \\ r(i-1, j-1) + 1 & \text{if } x_i = y_j \\ max\{r(i-1, j), r(i, j-1)\} & \text{if} x_i \neq y_j \end{cases}$$

$r(i, j)$ is determined by comparing the items $x_i$ and $y_i$: if they are same, then the length of $lcs(X_i, Y_j)$ equals $r(i-1, j-1)$ adds 1; if they are not same, then the length of $lcs(X_i, Y_j)$ equals the longer of the two longest common sequences $lcs(X_{i-1}, Y_j)$ and $lcs(X_i, Y_{j-1})$. Based on the recursive definition, the length of $lcs(X, Y)$ is found by a backtracking programming algorithm with time complexity $O(m*n)$. More algorithms are proposed with time complexity better than $O(m*n)$ [40, 41].

## 2.5 Extracting Representative Sequences

Frequent pattern mining and clustering are usually applied on the first steps to analyze a large-scale dataset, which helps produce categories and discover important features in defining the belonging of an instance to a category. Extracting representative is a task of summarizing a set of categories, which extracts a smallest possible number of representatives that ensure a given coverage of the whole category. Extracting representative is an important further analysis step to reduce intermediate preliminary patterns and to express categories.

"Representative set is defined as a set of non redundant typical patterns that largely, though not necessarily exhaustively covers the whole set" [42]. More specifically, the representative pattern extracting algorithm takes in the first-step data mining result, a similarity/distance measure method, a representativeness criterion, and a similarity/distance threshold under which two instances are considered as redundant or not [42].

Firstly, a list of candidate representatives are produces by computing representativeness score according to the selected criterion and similarity/distance measure. For example, if we consider selecting neighbourhood density as criterion, the representativeness score is computed according to the number of neighbourhood within a radius. If an instance has more neighbours, it is assigned a higher representativeness score. All instances are sorted in descending order, and an iteration process is performed on the ordered list to eliminate redundancy. The redundancy eliminating procedure is as follows: select the first instance into the representative list; then process each next instance in the ordered list of candidates; if this instance is not similar to anyone that has already been assigned into the representative list, add this instance into representative list. The iteration process stops when the selected sequences in representative list already satisfy the specified coverage threshold.

## 2.6 Z notation

Formal specification uses mathematical notations to precisely and clearly describe a computer system, which effectively helps to abstract and revise the system design. Formal specification is independent of system implementation using programming language, so that the system designer uses formal specification to describe what the system should do, rather than how the system should do it. Z notation is a formal specification language, which allows system designer uses mathematical data types, operators and expressions to model the data structure in a system. Another main flavor of Z notation is a way of splitting the specification into schemas to assist system designer in presenting the system piece in piece. A schema in Z is used to describe both static and dynamic aspects of system [11]. For example, the static aspects include the system states and invariant state transition matrix; and the dynamic aspects are possible operations and the state change when an operation happens.

Consider a system of checking hotel availability, we need to deal with customer's names and with reserved rooms. So we introduce the set of people names and the set of all rooms of the hotel as the basic types of the system specification, namely people and rooms:

$$[NAME, ROOM]$$

The state of the system is defined as following schema, which describes system static aspects of what kind of objects the types can contain. The part above the central dividing line declares some variables, and the part below the central dividing line provides relationships between the values of the variables which is always true in every state of the system and is maintained by every operation on it. Two variables are defined: *customer* is a set of people names who have already made reservation; and *reserved* is a function which, when a certain people names are applied, gives the rooms reserved for them.

$$
\begin{array}{|l}
\hline
\textit{HotelReservation} \\
\textit{customer} : \mathbb{P}\,\textit{NAME} \\
\textit{reserved} : \textit{NAME} \pfun \textit{ROOM} \\
\hline
\textit{customer} = \operatorname{dom} \textit{reserved} \\
\hline
\end{array}
$$

An operation of adding a new record of room reservation is defined as following schema, which describes system dynamic aspects of an operation. The declaration $\Delta \mathit{HotelReservation}$ alerts that the schema is describing a state change. $\mathit{name}?$ and $\mathit{room}?$ are two inputs of the operation $\mathit{AddReservation}$; $\mathit{customer}$ and $\mathit{reserved}$ are observations before the operation; and *customer'* and *reserved'* are observations after the operation which indicates an empty room is reserved by a new customer now.

$$
\begin{array}{|l}
\hline
\textit{AddReservation} \\
\Delta \textit{HotelReservation} \\
\textit{name}? : \textit{NAME} \\
\textit{room}? : \textit{ROOM} \\
\hline
\textit{name}? \notin \operatorname{dom} \textit{reserved} \\
\textit{customer}' = \textit{customer} \cup \{\textit{name}?\} \\
\textit{reserved}' = \textit{reserved} \cup \{\textit{name}? \mapsto \textit{room}?\} \\
\hline
\end{array}
$$

## 2.7 Process Algebra and BPQL

Algebra is a formal symbolic language, which investigates the relations and properties of numbers by means of general symbols. Process algebra is an algebra approach to the study of concurrent processes, concentrating on the behavior of communication. In many complex concurrent systems like network protocols and biology, process algebra is widely used for representing and reasoning about such systems in mathematical manner [43]. Process algebra is widely considered in the theory community but is not well known in the field of user behavior analysis. The drawback of specification represented by process algebra is not intuitive and hard to understand, especially for system administrators without enough mathematics and theory knowledge [44]. Thus we designed an intuitive BPQL (Behavior Pattern Query Language) to abstract high-level patterns, which allows system administrators to express their needs and somehow control the algorithm focusing on what is really interesting. Compared with learning process algebra, BPQL is easy to understand by system administrators because: i) employing intuitive and readable key words and syntax; ii) particularly designed for user behavior patterns which only includes a small set of symbols; and iii) deep integration with proposed behavior pattern mining algorithm.

## 2.8 R Packages

R is a programming language and open source software environment for statistical computing and graphics [45]. It provides a wide variety of data mining techniques and graphical facilities, such as time-series analysis, clustering and classification. R can be easily expended via user-created packages. An R package includes a set of powerful functions, which enables data analysts to go deep data mining by writing only several lines of code. "CRAN Task Views" [46] is a directory of links to available packages in wide range of research topics, such as machine learning, high performance computing, natural language processing, and medical image analysis. Some of R packages are used in developing toolkit for this thesis: "arules" is an R-package that provides the infrastructure for representing, manipulating and analyzing frequent itemset patterns and association rules among transaction data [47]; "arulesSequences" implements mining frequent sequence pattern algorithms [48]; "cluster" package in R implemented a variety of algorithms for cluster analysis [49]; and "TraMineR" is a toolbox for mining, describing and visualizing sequences of events [50].

# Chapter 3

# Behavior Pattern Mining

This chapter introduces an abstract behavior model, which is generic, system independent and configurable based on the target application domain. A behavior pattern is defined as: consistent observations of a sequence of actions that a user or a group of users conducted in a common context during a specific time interval (e.g., an hour, a day, a week). An abstract behavior model is introduced to represent user common behavior pattern. An interactive and iterative method based on several data mining techniques is proposed that can efficiently discover user's common behavior pattern from unlabeled dataset.

This chapter is organized as follows. Section 3.1 presents a dry run testing example to illustrate the maximal association relationship and user behavior pattern mining techniques. Section 3.2 defines an abstract behavior model and the basic concepts that are employed in the proposed model. Section 3.3 explains the behavior pattern mining process. Section 3.4 introduces a behavior pattern query language that allows analysts to describe the high level and abstract event clusters for grouping highly related events.

Table 3.1: An example of event database in medical imaging system

| Id | Date | Time | User | Role | Location | Operation | Resource |
|---|---|---|---|---|---|---|---|
| 1 | Mon | 10:00 | John | Radiologist | CT Lab | Order Exam | CT on Chest |
| 2 | Mon | 10:10 | John | Radiologist | CT Lab | Search&View | History Exams |
| 3 | Mon | 10:30 | John | Radiologist | CT Lab | Take Exam | CT on Chest |
| 4 | Mon | 10:50 | John | Radiologist | CT Lab | Read | Demographic Data |
| 5 | Mon | 11:20 | Emma | Radiologist | X-Ray Lab | Read | Diagnose Report |
| 6 | Mon | 11:30 | Emma | Radiologist | X-Ray Lab | Order Exam | X-Ray On Eye |
| 7 | Mon | 11:35 | Emma | Radiologist | X-Ray Lab | Search&View | History Exams |
| 8 | Mon | 11:45 | Emma | Radiologist | X-Ray Lab | Take Exam | X-Ray On Eye |
| 9 | Mon | 15:00 | Philip | Physician | Office | Read | CT on Chest |
| 10 | Mon | 15:15 | Philip | Physician | Office | Search&View | History Exams |
| 11 | Mon | 15:45 | Philip | Physician | Office | Create | Diagnose Report |
| 12 | Tue | 9:00 | John | Radiologist | CT Lab | Order Exam | CT on Lung |
| 13 | Tue | 9:05 | John | Radiologist | CT Lab | Search&View | History Exams |
| 14 | Tue | 9:15 | John | Radiologist | CT Lab | Take Exam | CT on Lung |
| 15 | Tue | 10:00 | Emma | Radiologist | X-Ray Lab | Order Exam | X-Ray On Brain |
| 16 | Tue | 10:10 | Emma | Radiologist | X-Ray Lab | Search&View | History Exams |
| 17 | Tue | 11:00 | Emma | Radiologist | X-Ray Lab | Take Exam | X-Ray On Brain |
| 18 | Tue | 14:00 | Philip | Physician | Office | Read | CT on Eye |
| 19 | Tue | 14:10 | Philip | Physician | Office | Search&View | History Exams |
| 20 | Tue | 14:30 | Philip | Physician | Office | Create | Diagnose Report |

## 3.1 Motivating Example

Table 3.1 demonstrates an example event database from medical imaging systems [51, 52] that trace three users' access to the system resources. User *John* and *Emma* are radiologist; user *Philip* is physician. Two typical imaging workflows are seen from this event database: *radiology-workflow*, including events 1, 2, 3, where the radiologist takes a new examination for a patient; and *diagnostic-workflow*, including events 9, 10, 11, where the physician writes a diagnostic report for a new examination. The radiology-workflow process is as follows: i) radiologist acquires an order of examination from pending order list; ii) she usually reviews the history images of the patient before taking examination; and iii) she takes the examination for the patient. Examination results (medical images) can be accessed for diagnosis from different locations. The diagnostic-workflow process is as follows: i) physician selects and views a new image of the patient; ii) he reviews patient's history examinations; and iii) he creates a diagnosis report for the new examination.

Table 3.2: Attribute representation used in the example

| **Attribute Name** | **Attribute Values** | **Attribute Encoding** |
|---|---|---|
| Event Id | Incremental integer starting from 1 | E-1, E-2, E-3,..., E-n |
| Date | Monday, Tuesday | D-1, D-2 |
| Time | Morning (00:00-12:00)<br>Afternoon(12:00-24:00) | T-1<br>T-2 |
| User | John, Emma, Philip | U-1, U-2, U-3 |
| Role | Radiologist, Physician | R-1, R-2 |
| Location | CT Lab, X-Ray Lab,<br>Office | L-1, L-2<br>L-3 |
| Operation | Order an Exam, Search & View, Take Exam,<br>Read, Create | O-1, O-2, O-3<br>O-4, O-5 |
| Resource Type | Image, History Exams,<br>Diagnose Report, Demographic Data | S-1, S-2<br>S-3, S-4 |

Let us now assume that we want to extract the common behavior patterns among different users as follows. I) What are the frequent actions? II) Are these frequent actions always performed in order? III) Is there any common context when the user performs the actions? IV) How often does the user perform the same actions?

### 3.1.1 Event Representation

As the first step of data analysis, raw dataset is parsed and represented. Each event is represented as a group of attributes. Attribute names, values and encoding used in the example are shown in Table 3.2. The encoded event database used in mining operation is shown in Table 3.3.

Table 3.3: Transformed event database used in mining operation

| Id | Date | Time | User | Role | Location | Operation | Resource |
|---|---|---|---|---|---|---|---|
| E-1 | D-1 | T-1 | U-1 | R-1 | L-1 | O-1 | S-1 |
| E-2 | D-1 | T-1 | U-1 | R-1 | L-1 | O-2 | S-2 |
| E-3 | D-1 | T-1 | U-1 | R-1 | L-1 | O-3 | S-1 |
| E-4 | D-1 | T-1 | U-1 | R-1 | L-1 | O-4 | S-4 |
| E-5 | D-1 | T-1 | U-2 | R-1 | L-2 | O-4 | S-3 |
| E-6 | D-1 | T-1 | U-2 | R-1 | L-2 | O-1 | S-1 |
| E-7 | D-1 | T-1 | U-2 | R-1 | L-2 | O-2 | S-2 |
| E-8 | D-1 | T-1 | U-2 | R-1 | L-2 | O-3 | S-1 |
| E-9 | D-1 | T-2 | U-3 | R-2 | L-3 | O-4 | S-1 |
| E-10 | D-1 | T-2 | U-3 | R-2 | L-3 | O-2 | S-2 |
| E-11 | D-1 | T-2 | U-3 | R-2 | L-3 | O-5 | S-3 |
| E-12 | D-2 | T-1 | U-1 | R-1 | L-1 | O-1 | S-1 |
| E-13 | D-2 | T-1 | U-1 | R-1 | L-1 | O-2 | S-2 |
| E-14 | D-2 | T-1 | U-1 | R-1 | L-1 | O-3 | S-1 |
| E-15 | D-2 | T-1 | U-2 | R-1 | L-2 | O-1 | S-1 |
| E-16 | D-2 | T-1 | U-2 | R-1 | L-2 | O-2 | S-2 |
| E-17 | D-2 | T-1 | U-2 | R-1 | L-2 | O-3 | S-1 |
| E-18 | D-2 | T-2 | U-3 | R-2 | L-3 | O-4 | S-1 |
| E-19 | D-2 | T-2 | U-3 | R-2 | L-3 | O-2 | S-2 |
| E-20 | D-2 | T-2 | U-3 | R-2 | L-3 | O-5 | S-3 |

### 3.1.2 Maximal Association

Finding highly related event groups is necessary for discovering user common behaviors, especially in the case of large dataset. An example might be that a group of events about people working at radiology department start viewing medical images between 9:00am to 10:00am. The patterns discovered from this group of events represent the common behavior of radiology department at specific time. Finding all such highly related event groups is valuable to system administrators to categorize system user behaviors. Without prior knowledge, we apply association mining on the encoded event database for discovering knowledge of the dataset such as the most associated events and the interesting attributes. It is intended to bring together highly related events, and the shared attributes indicate the common context of the group of events.

We define *maximal association* in a group of events in the form of a maximum set of events that all share the same set of attribute values. No larger set of events can be found that share all these attribute values, and vice-versa. We refer to the group of events as the "*container*" and the shared set of attribute values as the "*itemset*" (each attribute value is viewed as an item). In this sense, the container of events and the itemset are denoted as a maximal association group (MAG). We aim to find large MAGs where both the containers and itemsets are large. Association mining algorithm [19, 53, 54] is applied on the encoded event database with minimum support 8 (the value is adjusted based on mining result), where the support of an itemset is how many times the itemset appears in the event database. Finally the algorithm extracts 9 frequent itemsets. We remove the MAGs whose itemsets are included in a proper superset that is also an itemset in another MAG. This phase deletes a large number of redundant MAGs. Figure 3.1 illustrates the extracted 5 MAGs after the pruning phase. MAGs have overlaps since both an event and an attribute may belong to more than one MAG. For

| MAG | Itemset | Container |
|---|---|---|
| 1 | {D-1} | {E-1, E-2, E-3, E-4, E-5, E-6, E-7, E-8, E-9, E-10, E-11} |
| 2 | {D-2} | {E-12, E-13, E-15, E-16, E-17, E-18, E-19, E-20} |
| 3 | {T-1, R-1} | {E-1, E-2, E-3, E-4. E-5, E-6, E-7, E-8, E-12, E-13, E-15, E-16, E-17} |
| 4 | {s-1} | {E-1, E-2, E-3, E-6, E-8, E-12, E-13, E-14, E-15, E-17} |
| 5 | {T-1, R-1, S-1} | {E-1, E-3, E-6, E-8, E-12, E-14, E-15, E-17 } |

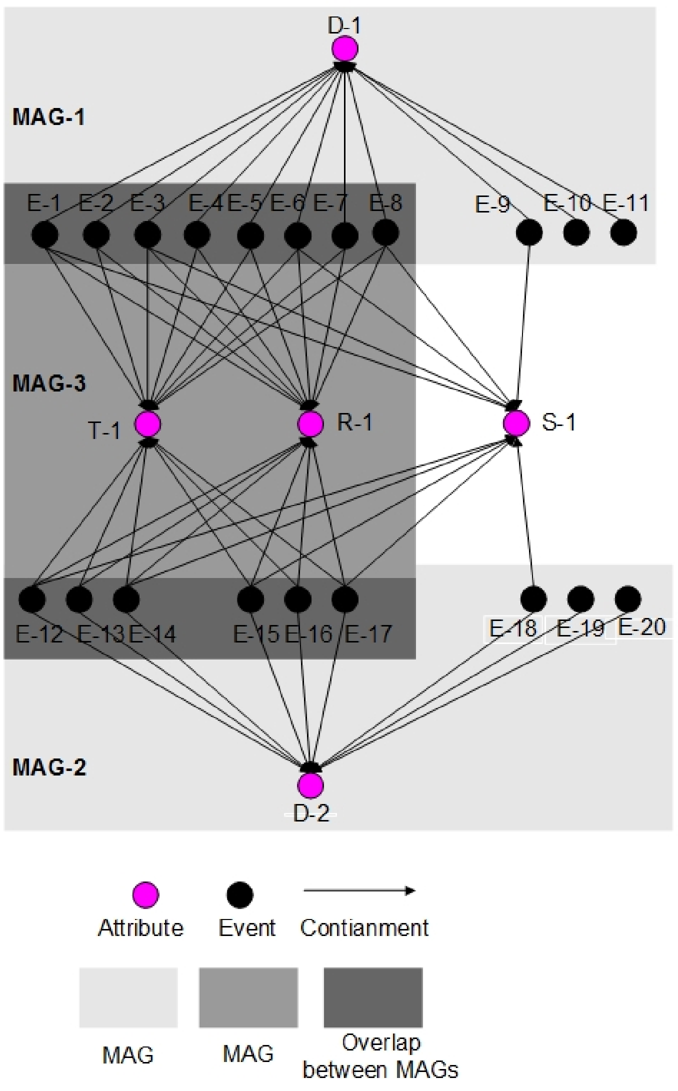


Figure 3.1: Maximal association groups extracted from encoded event database

example, we can see from Figure 3.1 that: events {E-1, E-2, E-3, E-4, E-5, E-6, E-7, E-8} exist in "*MAG-1*" and "*MAG-3*"; and attributes {T-1, R-1} appear in "*MAG-3*" and "*MAG-5*".

On the assumption that an event is more significant if it is associated with a large number of events through sharing more common attributes, each MAG represents a group of similar events and is considered as the building block for the proposed behavior pattern mining approach. An association-based similarity between events inside a MAG and between two MAGs are defined in section 4.2, which encodes the size and structure of MAG. The association-based similarity measure result provides analysts a clue to the significance of events and the common characteristics among such events. Considering both itemset length and container size, "*MAG-3*" might receive the highest similarity value since its itemset length is "2" and the size of container is "14". The analyst may assign all events in "*MAG-3*" into a cluster to study the common behavior of radiologist in the morning. "*MAG-1*" with itemset {D-1} is close to "*MAG-3*" as they share 8 events with two same attribute values totally, and "*MAG-2*" with itemset {D-2} is also close to "*MAG-3*" as they share 6 events with two same attribute values. The analyst may assign events in "*MAG-1*" and "*MAG-3*" into the first cluster to study the common behavior of radiologist in the morning of the first day, and assign events in "*MAG-2*" and "*MAG-3*" into the second cluster to learn if any change can be seen from the common behavior the very next day. These clusters of events are collected for subsequent behavior pattern mining phase.

In general, the number of shared attributes contributes more on the closeness of the events than the number of sharing events, if a group of events are considered for their similarity. Moreover, the attributes may be assigned different weight when computing association measure, such as attribute "*role*" contributes more than attribute "*accessed resource*". The detailed association computing is explained in Chapter 4.

By decreasing the minimum support value during the association mining, we may find a large number of MAGs and construct a ranking list of significant MAGs with rich associations. To inject data analyst's domain knowledge into the analysis process, and also to reduce the search space in subsequent behavior mining phase, we propose a behavior pattern query language that allows analysts to express their interest using cluster constraints. For example, the analyst knows there are two labs in radiology department, and he might want to know what are the common behavior at different labs in the morning. The behavior pattern query language will be discussed in Section 3.4.

### 3.1.3 Sequential Pattern Mining

In order to mine frequent behavior patterns under specific context, we apply sequential pattern mining on each constrained cluster to discover frequent action-sequences. An "*action*" denotes to one or more attributes extracted from one event. For example, an action can be: a user is working at location "L-1"; or a user accesses a patient's image "O-4, S-1". An "*action sequence*" is an ordered list of actions. For example, two action sequences can be: i) a user works daily at two locations with the order "<<L-1>, <L-2>>"; and ii) a user in a workflow performs the following operations "<<O-1, S-1>, <O-2, S-2>,<O-3, S-1>>".

A "*user-sequence*" is an ordered list of events. The problem of mining behavior pattern of a user is to find the action-sequences that appear frequently within one user's sequence; identifying common behavior patterns among multiple users is mining action-sequences that appear frequently among all user-sequences. Figure 3.2 shows the extracted behavior patterns from associated event-group "*MAG-3*" with common context <T-1, R-1>. Both "U-1" and "U-2" perform the following action sequence: order an examination "<O-1, S-1>", search and view history

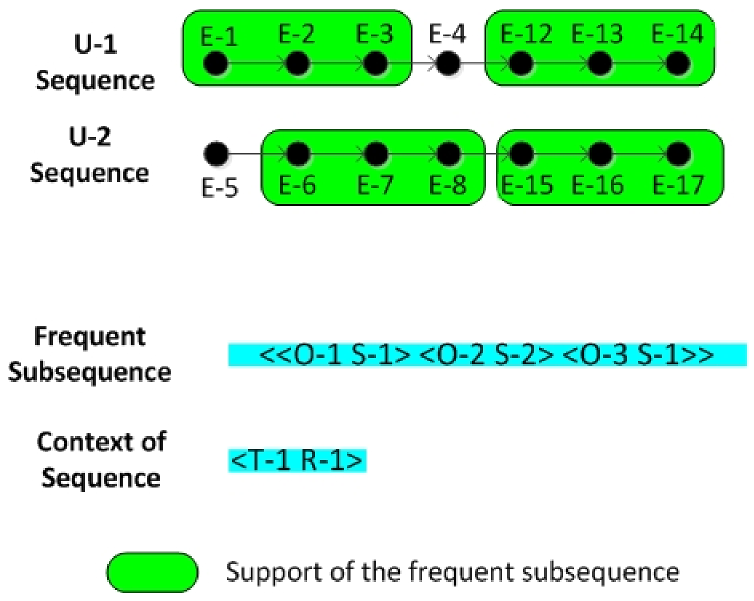


Figure 3.2: Extracted behavior patterns of radiology-workflow: subsequence <<O-1, S-1>, <O-2, S-2>,<O-3, S-1>> appears both in U-1 sequence and U-2 sequence. <E-1, E-2, E-3> contains <<O-1, S-1>, <O-2, S-2>,<O-3, S-1>> as <O-1, S-1> in E-1, <O-2, S-2> in E-2, and <O-3, S-1> in E-3.

examinations of the patient "<O-2, S-2>", and then take the examination "<O-3, S-1>". This action sequence reflects the behavior of taking radiology-workflow. The common context attributes describe circumstances for the complete sequence. It means the radiology-workflow always happens in the morning and with assigned role "radiologist".

The support of an action-sequence is defined as the appearance of such action-sequence in user-sequences. So the support of the action sequence in our example is 2: <E-1, E-2, E-3> and <E-12, E-13, E-14> in U-1's sequence; <E-6, E-7, E-8> and <E-15, E-16, E-17> in U-2's sequence. The support of an subsequence only count one occurrence in a user sequence even though the subsequence appears twice in user's sequence. The analyst may divide the user sequence into several time constrained user sequences (e.g., the duration of a time constrained user sequence cannot exceed 10 minutes, an hour or a day). In this case, the support of a subsequence may count more than once if they are split into separated time-constrained user sequences.

## 3.2 Behavior Representation

The following statements are sample user-system interactions that can be expressed as the user behavior patterns:

- User $u_i$ in a workflow process, reads resources $r_a$ and $r_b$ that belong to owner $o_l$, and then updates resource $r_b$.
- User $u_i$ in a day takes roles in the following order: $r_k$, $r_l$ ,$r_h$.
- User $u_i$ in a week works at $loc_a$ from Monday to Friday, and works at location $loc_b$ on Saturday.
- User $u_i$ accesses resources $r_a$, $r_b$ about $n_1$ times on average between times $t_1$ and $t_2$.

In the above examples, typical user behavior patterns can be expressed using association pattern, sequence pattern and timing rules: association enforces that the occurrence of one value should result in one or more other values (e.g., specific users normally work at specific locations); sequence reflects the continuous or connected series of actions (e.g., what a user does next is impacted by what he or she did in the past few steps); and time rules indicate the frequency that a behavior repeats itself.

We define the basic concepts that are employed in the proposed model.

**Event**

A single user-system interaction (i.e., any communication with the system such as image storage, retrieval, query, etc.) is recorded as an *event*. An event is extracted from audit log and represented by a set of attribute values. Whenever any attribute of the event changes, a new event is recorded.

**Attribute**

*Attribute* is the extracted characteristic from application domain. A set of domain specific attributes are fundamental elements that can be ascribed to events.

An attribute is represented by a name and a domain of attribute values. For example, "operation", "location" and "service" are generic attributes extracted from the audit logs. "Read exam" and "update report" are values of attribute "service" in the medical domain. The attributes normally can be classified into three categorizes:

- *Actor* attributes are used to explain the subject of events. For example, *User* is an actor attribute, which identifies an individual who performs the action; *Role* is also an actor attribute which determines a group of people having similar privileges and responsibilities.
- *Contextual* attributes determine the context (or neighborhood) of events. For example, *Location* can be a contextual attribute which limits the neighbor events happened at the same location or nearby; *Time* can be considered as a contextual attribute which determines the neighbor events happened within a short period of time; *Patient* could be a contextual attribute which explains the neighbor events should be accessing the health records of a specific patient.
- *Behavioral* attributes define the non-actor and non-contextual characteristics of the events. For example, *Action* is a behavioral attribute, which describes one step of the workflow under a specific scenario; *Location* can also be a behavioral attribute which indicates one location of ward-round by nurses. Behavioral attributes in a dataset may be contextual attributes in another dataset, for example, *location* is a behavioral attribute in robot moving dataset but a contextual attribute in service accessing dataset.

#### **User Behavior**

*User behavior* is extracted from a collection of user-system interactions (i.e., events). This thesis proposes a user behavior pattern representation based on association, sequencing and timing rules. Association indicates the concurrence of a set of attribute values together. Sequencing requires that a series of steps occur in a certain order. Timing allows sequencing the events; limits the events occurrence frequency; and assigns the gaps between events. A user behavior is represented as a quadruple:

$Behavior =< Actor, Sequence, Context, TimeInterval >$

where *Actor* issues a behavior; *Sequence* is the sequence of steps performed by the Actor; *Context* is the circumstances in which the behavior takes place; and *Time Interval* is the time duration within which the behavior is recovered.

**Common Behavior**

Intuitively, frequently occurring user behaviors that are discovered from a large event repository are reasonable to be regarded as user common behaviors. In other words, if a specific behavior is repeatedly performed by a group of users, most probably it is a common behavior. Also, given a large repository of events, we can expect to discover a collection of common behaviors. The actor of a behavior is extracted from the actor attributes of events to categorize the behaviors. The context of a behavior is extracted from contextual attributes of the events to determine the neighborhood. The sequence of a behavior is extracted from behavioral attributes to explain user's behavioral characteristics. The time interval of a behavior is extracted from the time constraints.

Figure 3.3 illustrates the UML class diagram of the proposed behavior model, which defines different types of entities for describing behavioral elements.

- Class *Attribute* includes attribute name and attribute value that are extracted from audit log. Class *Event* includes a list of attributes and an unique ID.

- Class *MAG* is an aggregation of class *Event*. Events that share a set of attribute values (named as itemset) are assigned to a maximal association

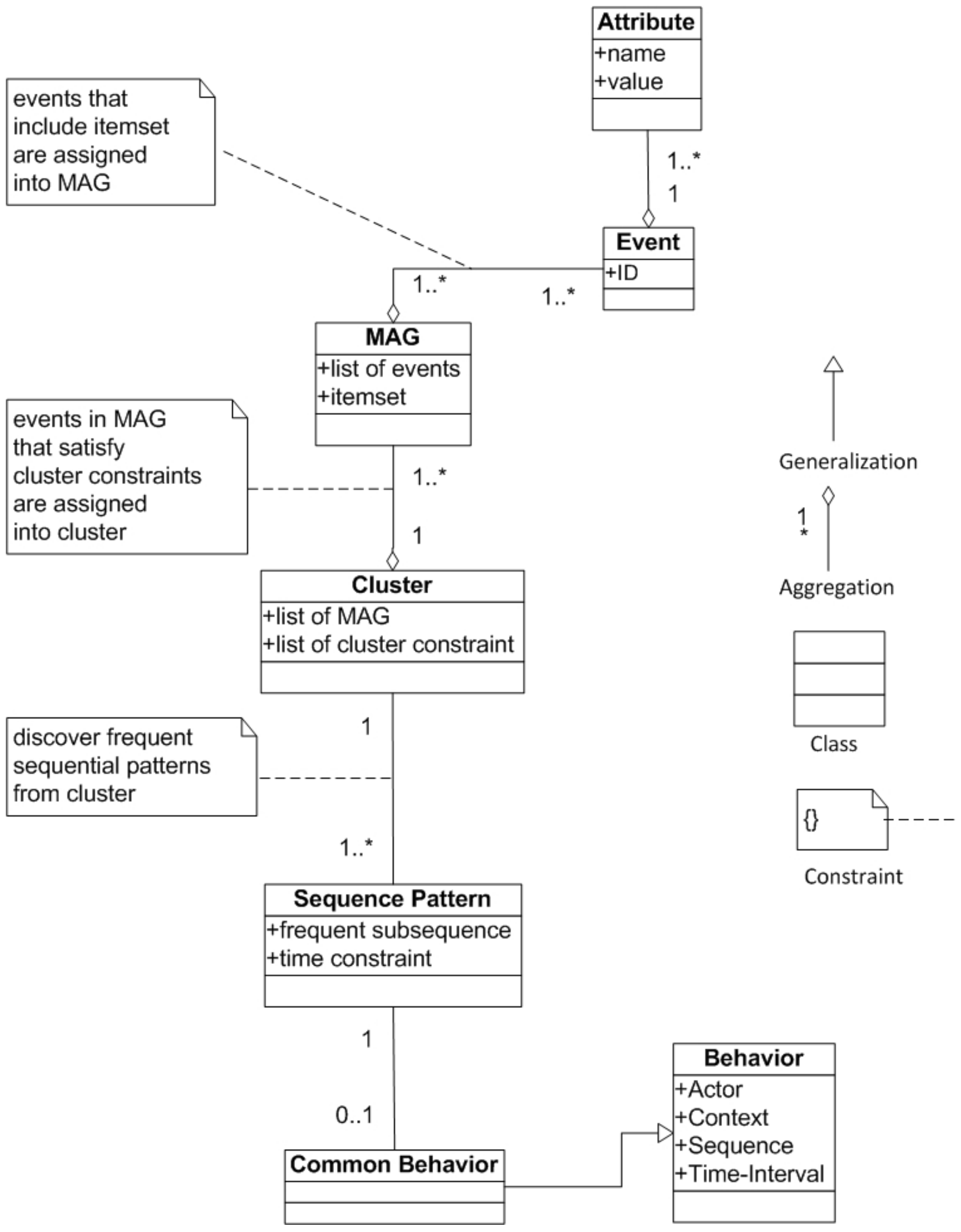


Figure 3.3: Class diagram of an abstract behavior model suitable for user behavior pattern mining task

group *MAG*. An event may be assigned to more than one MAG as it shares different itemset with different group of events.

- Class *Cluster* is an aggregation of class *MAG*. One or more *MAG* instances construct the search domain for constrained clustering. An *Event* in a source *MAG* is assigned to a *Cluster* if it satisfies the cluster constraint. Cluster constraint restricts the cluster size and expresses the analyst's focus. Cluster constraint is defined by the analyst using significant attribute values and it is either an "intra-cluster-constraint" (defined for one cluster) or an "inter-cluster-constraint" (defined between two clusters).

- Class *Sequence Pattern* is frequent subsequence pattern extracted from *Cluster* which is the result of sequential pattern mining operation. Attribute *subsequence* is a discovered sequential pattern occurring frequently in *Cluster*. Attribute *time constraint* limits the maximum duration of *Sequence Pattern*.

- Class *Behavior* contains some common characteristics among events in *Cluster*. Its attributes are designed to describe the four intergradient of a behavior: *Actor* issues a behavior where an actor can be an individual person or a group of people; *Sequence* represents the activities that an actor performs on the system; *Context* indicates the environment in which a behavior usually happens; and *Time-Interval* describes the duration and frequency of the behavior.

- Class *Common Behavior* is converted from *Sequence Pattern*, where *subsequence* maps to the behavior's *Sequence*; *itemset* and *cluster constraint* feeds behavior's *Context*; and *time constraint* links to behavior's *Time-Interval*.

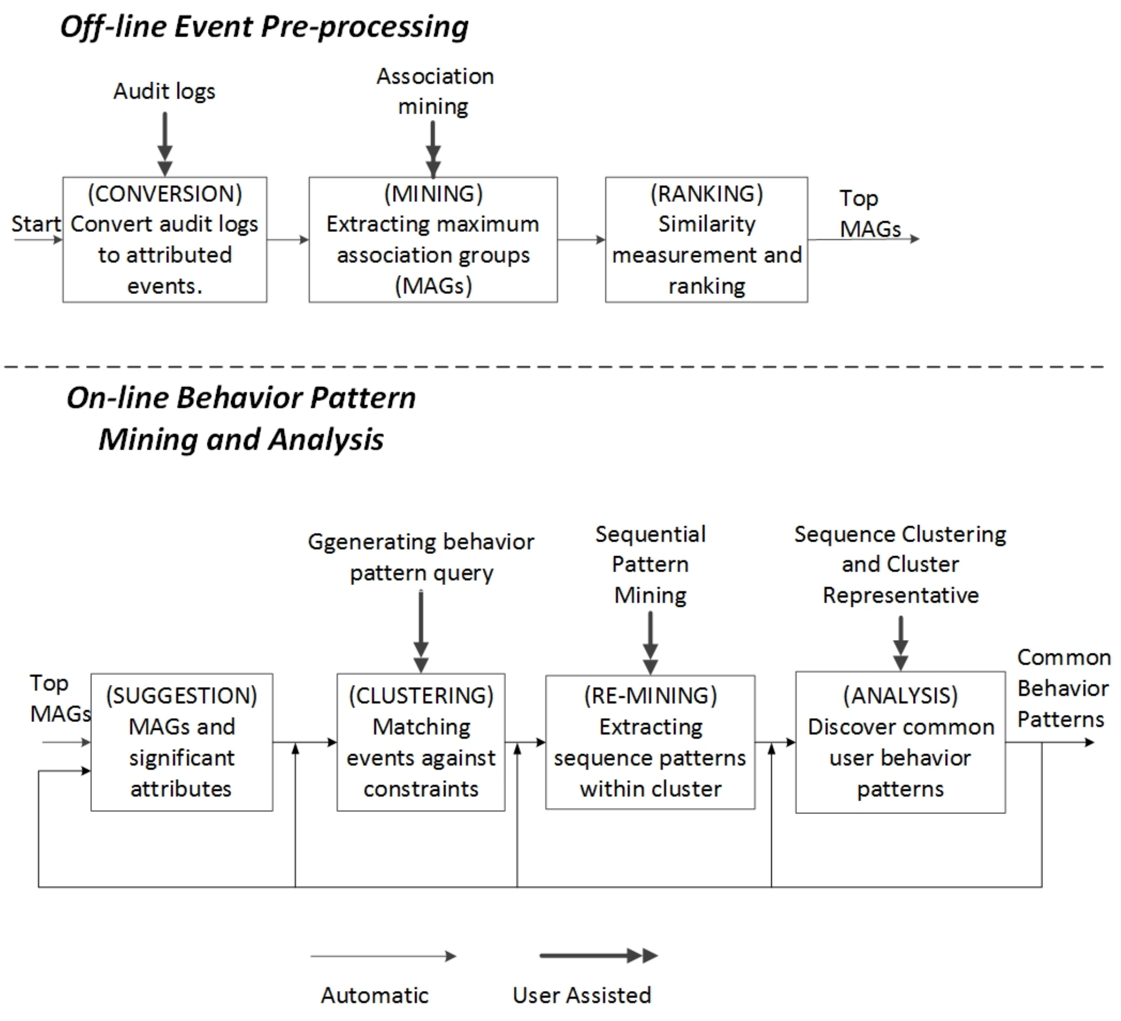

Off-line Event Pre-processing
Audit logs
Association mining
Start
(CONVERSION) Convert audit logs to attributed events.
(MINING) Extracting maximum association groups (MAGs)
(RANKING) Similarity measurement and ranking
Top MAGs
On-line Behavior Pattern Mining and Analysis
Ggenerating behavior pattern query
Sequential Pattern Mining
Sequence Clustering and Cluster Representative
Top MAGs
(SUGGESTION) MAGs and significant attributes
(CLUSTERING) Matching events against constraints
(RE-MINING) Extracting sequence patterns within cluster
(ANALYSIS) Discover common user behavior patterns
Common Behavior Patterns
Automatic
User Assisted

Figure 3.4: Proposed user common behavior pattern recovery process: i) off-line preprocessing discover significant group of events to reduce search space; ii) online-behavior pattern mining and analysis interactively and iteratively extract common behavior patterns using data mining techniques and domain knowledge.

## 3.3 Behavior Pattern Recovery Process

Figure 3.4 illustrates the proposed knowledge-driven behavior pattern discovery process and the techniques used in different stages.

**Step 1, Off-line Event Pre-processing.** The audit logs from the target distributed system are collected, parsed and converted into encoded attributed events. A data mining engine applies association mining operation on the events to extract a large number of highly related MAGs, where each MAG consists of

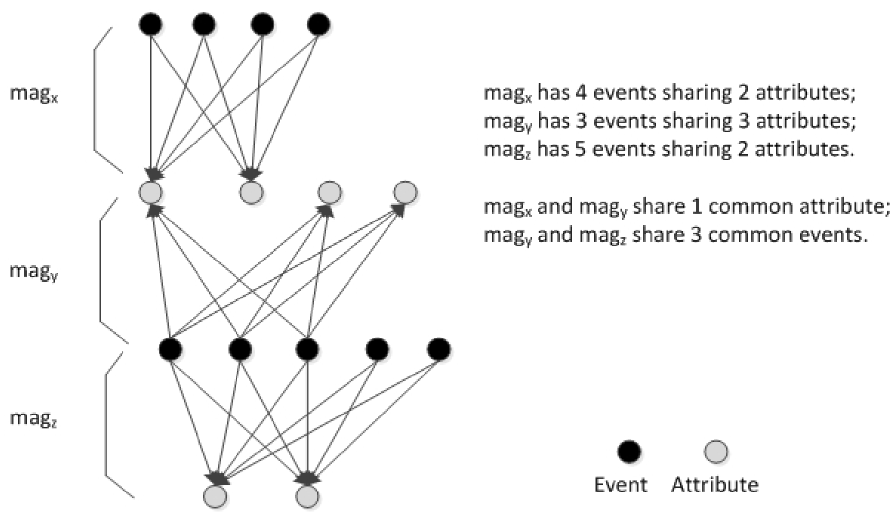

$mag_x$
$mag_y$
$mag_z$
$mag_x$ has 4 events sharing 2 attributes;
$mag_y$ has 3 events sharing 3 attributes;
$mag_z$ has 5 events sharing 2 attributes.
$mag_x$ and $mag_y$ share 1 common attribute;
$mag_y$ and $mag_z$ share 3 common events.
Event
Attribute

Figure 3.5: Maximal association groups: $mag_x, mag_y, mag_z$

a maximum set of events that all share a maximum set of attribute values. An event may be assigned to more than one MAG as it shares different attribute values with different group of events. Figure 3.5 presents 3 maximal association groups $mag_x, mag_y$ and $mag_z$: $mag_x$ and $mag_y$ have overlap of sharing common attributes, and $mag_y$ and $mag_z$ have overlap of sharing common events. Each MAG represents highly similar events and is considered as the building block for the proposed behavior recovery approach.

We define an association-based similarity metric between events in the same MAG, which encode both the size and the structure of a MAG. On the assumption that the event is more significant if it is associated with a large number of events through sharing more common attributes, such a similarity metric is used in generating constrained-clusters of events for subsequent behavior pattern mining.

Association is a strong constraint which may produce a large number of small MAGs where very limited number of events are assigned to each MAG. Few behavior pattern is extracted from such small MAG. To address this issue, we also define an association-based similarity metric to measure the closeness between two MAGs. With the result of similarity between any two MAGs, the analyst is allowed to select several close MAGs as the source of constrained-cluster.

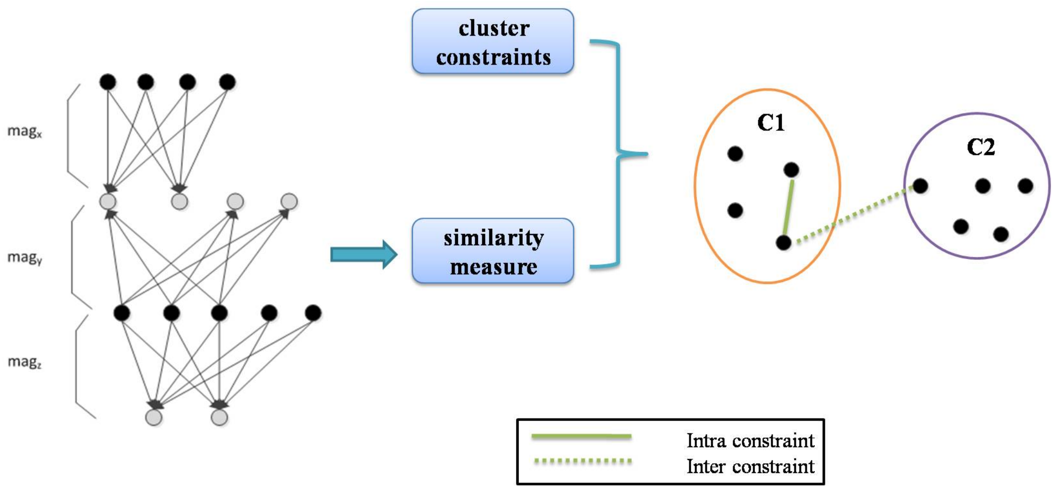


Figure 3.6: Constrained event clusters: $C1$, $C2$

In order to restrict the search space during populating the clusters, we create a database of MAGs. This database will be used for four purposes: i) for ranking MAGs based on association measure between events of one MAG to provide recommendations for highly related group of events; and ii) for collecting close MAGs based on association measure between two MAGs to couple small MAGs; iii) for providing attribute statistics which allows the analyst to focus on the significant feature in composing the behavior pattern query; and iv) as restricted search space to collect events satisfying cluster constraints in online behavior pattern mining phase.

**Step 2, On-line Behavior Pattern Mining.** Using an iterative process, the user incrementally selects one or more MAGs from the ranking list; and generates a BPQ (Behavior Pattern Query) based on the suggested attributes (e.g., shared attributes among most events in a MAG) and user's domain knowledge to collect highly related events from source MAGs. A link-constraint is defined using attribute values to restrict the events that are assigned to the clusters, which is either an "intra-constraint" (defined for one cluster) or an "inter-constraint" (defined between two clusters). The main purpose of such constraints is to focus on specific attribute values that the user is interested to investigate. The inter-constraints allow separating the clusters from each others; then the analyst can convert the inter-constraints into intra-constraints during the searching operation. Figure 3.6 presents the constrained event clustering, where similarity is measured based on association extracted from MAGs; *intra-constraint* specifies the conditions that instances in the same cluster should satisfy; and *inter-constraint* specifies the conditions that two instances from different cluster should satisfy.

The Sequential pattern mining is applied on each constrained-cluster to extract a number of frequent action-sequences. The sequential pattern mining algorithm is only controlled by one input parameter "*minsup*". If we make *minsup* too high, only very simple patterns are found, or none at all. But if we make *minsup* too low, the output includes a huge number of patterns, making it very hard to analyze and use the result. Sequential pattern mining may generate a huge number of short and trivial patterns but fail to extract interesting patterns approximately shared by a group of similar sequences. To address this issue, we applied clustering and representative extraction technique to summarize the large number of extracted frequent sequential patterns. A similarity-based clustering algorithm is applied on the sequential pattern mining result, which divides the frequent sequential patterns into a number of clusters. Clustering operation groups similar patterns in the same cluster but does not reduce the number of patterns. Then representative extraction algorithm is applied on each sequence cluster to eliminate redundancy from data of clusters. Representative sequences are a set of non-redundant typical sequences that largely cover the observed sequences in a cluster. Common behavior patterns are finally produced based on the extracted representative sequences.

A common action-sequence within the single user-sequence is that user's behavior. If a user's common action-sequence is also seen in other users' sequences, then it is common behavior of the system users within the constrained-cluster. The constraints of the cluster can be viewed as the common context of all representative sequences extracted from this cluster. Finally a number of common behavior

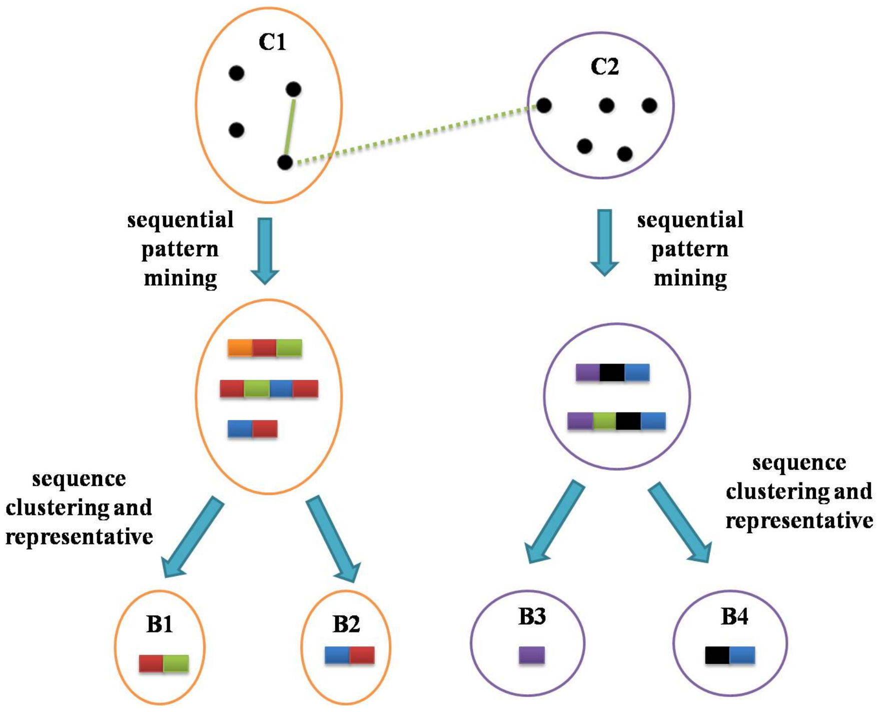

C1
C2
sequential pattern mining
sequential pattern mining
sequence clustering and representative
sequence clustering and representative
B1
B2
B3
B4

Figure 3.7: Common behavior extraction: $B1, B2, B3, B4$

patterns are extracted and represented as the quadruple: <actor, action-sequence, context, time interval>.

Figure 3.7 is an example of extracting common behavior patterns from constrained event clusters *C1* and *C2*. After applying frequent sequential pattern mining on *C1* and *C2*, some frequent subsequence patterns are discovered respectively. Sequence clustering and representative extraction are applied on extracted frequent subsequence patterns, and finally 4 behavior patterns are produced: *B1* and *B2* are common behavior patterns under the same context (*C1* cluster constraint); *B3* and *B4* are common behavior patterns under the same context(*C2* cluster constraint).

## 3.4 Behavior Pattern Query Language

Constraints are essential in many data mining applications for effectiveness and efficiency considerations, which allows the analysts to express their needs and somehow control the algorithm focusing on what is really interesting. In this thesis, we present a constraint-based Behavior Pattern Query Language (BPQL) which enables the analysts to define queries in a declarative way addressing desired patterns. These queries describe the high level and abstract clusters and their intra-cluster/inter-cluster constraints that allow the analysts to identify the system-user's behavior patterns that match with the defined high level constraints. The constrained clustering using BPQL is explained as follows:

- Each cluster is intended to be a loose search space for the sequential pattern mining (common behavior pattern mining).
- The clusters act as highly related groups of MAGs that satisfy particular properties such as all events that are highly related and also they are at *CT Lab in Radiology department*, or they are events that access certain type of resources, or happen at particular time-interval, or all related to certain roles, or a combination of the above. Therefore each cluster generates events that are related by association and also they satisfy user-defined properties.
- The other clusters can also gather the events with specific properties, and should satisfy the inter-cluster constrains (i.e., separation), which can be: the events that are at the other lab (e.g., X-ray Lab), or at different time-interval, or some property that is distinct from the earlier clusters.
- The event number of cluster is restricted to control the search space for common behavior pattern extraction, but should not be a strict value. The cluster should be large enough to gather almost all the events that satisfy the defined properties (both intra-cluster and inter-cluster constraints).
- The inter-cluster constraint will be translated into intra-cluster constraint by a pre-clustering operation. In a primitive version, the user performs this translation. For example, distance separation from the first cluster is

translated into events that happen in Toronto; or events that happen in a limited time-interval which are distinct form those in the first cluster.

- The constrained clustering is an important step for common behavior pattern mining, as it provides highly customizable and relevant search space.

Appendix A defines the keywords and syntax for the proposed BPQL. The following is a behavior pattern query example. The example explores behavior patterns during rush hour at specific location. In the attribute analysis phase, a number of maximal association groups are generated with association measure. For example, MAG *mag-1* is on the top of the ranking list of the association value within a MAG. And *mag-2* and *mag-3* have higher association value with *mag-1*. According to the statistics of MAGs, the analyst knows most of the events in *mag-1* occur in rush hours at location *CT Lab*. Then he may wish to investigate the difference between behavior in rush hours (from 9:00am to 11:00am) and behavior not in rush hours at *CT Lab*. The analyst selects *mag-1* as the source domain of cluster *C-1*; *mag-2* and *mag-3* as source domain of cluster *C-2* and *C-3*. Constrained-cluster *C-1* includes the events during rush hour at "*CT Lab*" from *mag-1*; and constrained-cluster *C-2* collects events before rush hour and constrained-cluster *C-3* collects events after rush hour at the same location. Inter-cluster constraints are converted to intra-cluster constraint, such as "*C-2.Location = C-1.Location; C-2.Time < C-1.Time*" is converted to *"Location = 'CT Lab'; Time < 9:00"* when selecting events to cluster. *SIZE-CONSTRAINT* specifies the preferred cluster size, which allows the analyst restrict the search space for further behavior pattern mining process. The behavior patterns extracted from cluster *C-1* is common behavior at *CT Lab* during rush hour, and behavior patterns extracted from cluster *C-2* and *C-3* are common behavior at *Lab CT* before rush hour and after rush hour respectively.

```
BEGIN-BPQ
    CLUSTER := C-1;
        MAG := mag-1;
        SIZE-CONSTRAINT := 5000;
        INTRA-CONSTRAINT
            Location = 'CT Lab';
            Time > 9:00; Time < 11:00;
    CLUSTER: C-2;
        MAG := mag-2, mag-3;
        SIZE-CONSTRAINT := 1000;
    CLUSTER: C-3;
        MAG := mag-2, mag-3;
        SIZE-CONSTRAINT := 1000;
    INTER-CONSTRAINT
        C-2.Location = C-1.Location;
        C-3.Location = C-1.Location;
        C-2.Time < C-1.Time;
        C-3.Time > C-1.Time;
END-BPQ.
```

# Chapter 4

# System Modeling

A formal software specification is a specification expressed in a language whose vocabulary, syntax and semantics are formally defined [55]. The specification language relies on mathematical concepts whose properties are well understood. The principle benefits of formal methods are in avoiding ambiguity and reducing the number of faults in system design. Model-based approach is one of the fundamental approaches of formal specification where a model of the system is built using mathematical constructs such as sets, sequences, system states, and the system operations that modify the system state [56]. Z specification is one of the widely used notations for developing model-based specification. This chapter models the user behavior pattern based security provisioning system by defining the data, types, functions, and relations using Z specification language.

This chapter is organized as follows. Section 4.1 formalizes common behavior pattern mining system using Z notation [11, 12]. Section 4.2 defines maximal association to measure the similarity between events in group. Section 4.3 defines the measure of the closeness of two sequences. Section 4.4 discusses the computational complexity of the proposed algorithms.

## 4.1 Formal Specification

Following are several definitions of Z specification [11] used in this thesis:

**Type Definition**

$$T ::= \ldots$$

A type definition introduces a new type that must not have a previous global declaration. $T$ on the left becomes a global identifier and its value is given by the expression on the right. Every set can be used as a type. An example of type is introducing of *NAME* and *ROOM* in the hotel reservation system in Section 2.6.

**Axiomatic Description**

$$\begin{array}{|l} \mathit{Declaration} \\ \hline \mathit{Predicate};\ \ldots;\ \mathit{Predicate} \end{array}$$

An axiomatic description introduces one or more new global variables (the part above the dividing line), and optionally specifies a set of constraints on their values (the part below the dividing line). *Predicate* specifies the constraints on the values of the global variables that are declared in *Declaration* or that have been declared previously.

**Schema Definition**

$Schema - Name$

$Ident? : ...$

$Ident! : ...$

$Declaration$

---

$Predicate; ...; Predicate$

Schema definition allows a system to be described separately, then related and combined. A single process in a complete system may be described in isolation using a schema definition, then related to the evolution of the system as a whole. A schema definition in this thesis introduces a new user-controlled process, where the process output varies by given different input. The word heading the box is the schema name; input variables are declared with "?" (e.g., *Ident?*); and output variables are declared with "!" (e.g., *Ident!*). The *Declaration* above the central dividing line declares some variables, and the *Predicate* gives a relationship between the values and variables declared above. Examples of schema are introducing the system state *HotelReservation* and an operation *AddReservation* in the hotel reservation system in Section 2.6.

Following introduces a high-level algorithm of user behaivor pattern mining, and formal specification using Z notation[11, 12].

Algorithm 1 accepts system audit log file as input to explore user common behavior patterns. The description of the algorithm with reference to its line numbers is as follows:

- *Line 1 to 3:* The audit logs are parsed, analyzed and converted into encoded attributed events. A data mining engine applies association mining operation [19] on the events to extract a set of maximum association groups (MAG), where each MAG consists of maximum set of events that all share

**Algorithm 1** User behavior pattern mining

```
INPUT: Audit Log File == $LOG$
OUTPUT: User Common Behavior Patterns == $B$
 1: Encoding&Attribute-Analysis: $LOG \Rightarrow$ Event Database == $E$
 2: Attribute-Association-Mining: $E \Rightarrow$ A set of Maximal Association Group == $MAG$
 3: BPQ&Constraint-Analysis: $MAG \Rightarrow$ Constrained Event Clusters == $CEC$
 4: for all constrained event cluster $CEC_i$ in set $CEC$ do
 5:     User-based-Sequencing: $CEC_i \Rightarrow$ User Sequence Clusters == $USC_i$
 6:     Time-Constrained-Analysis: $USC_i \Rightarrow$ Time-constrained User Sequence Clusters == $TCUSC_i$
 7: end for
 8: for all time-constrained user sequence cluster $TCUSC_i$ in set $TCUSC$ do
 9:     Sequential-Pattern-Mining: $TCUSC_i \Rightarrow$ Frequent Subsequence Patterns == $FSP$
10:     Similarity-based-Sequence-Clustering: $FSP \Rightarrow$ Frequent Subsequence Pattern Clusters == $FSPC$
11:     for all frequent subsequence pattern cluster $FSPC_j$ in set $FSPC$ do
12:         Representative-Sequence-Mining: $FSPC_j \Rightarrow$ Set of representative sequences == $RS$
13:         User Common Behavior Pattern Discovery: RS $\Rightarrow$ B
14:     end for
15: end for
```

a maximum set of attribute values. Each MAG represents highly similar events and is considered as the building block for constrained event clusters. The analyst selects one or more MAGs as source domain for event cluster operation, and generates cluster constraints based on the suggested attributes and domain knowledge to collect a large group of events that are highly or loosely related into constrained event clusters. A high-level behavior pattern query is explained in Section 3.4, which results in discovering more valuable user behavior patterns with limited search space in a reasonable execution time.

- *Line 4 to 7:* To explore user behavior patterns, the constrained event clusters are converted into user sequence clusters where each user sequence is a list of events performed by the same user. Without time restriction, the length of user sequence is exploding as the duration of the user sequence may be months even years depending on the collected audit logs. To address the issue of sequence length explosion, a user sequence is split into a number of time constrained user sequences with a maximum duration of a session, a day, or a week.

- *Line 8 to 9:* The frequent sequential pattern mining [20] is applied on each constrained user sequence clusters to extract a number of frequent subsequences. If a subsequence in a user sequence is also seen in other user sequences, then the subsequence will probably be an abstract of common behavior among system users within the constrained cluster.

- *Line 10 to 13:* A huge number of frequent subsequence patterns may be generated in the step of sequential pattern mining, which is very hard to be analysed. After dividing the frequent subsequences into a number of clusters according to sequence similarity, a representative pattern mining operation is applied on each subsequence cluster. These small amount of representative patterns are the baseline to construct the common user behavior patterns.

Table 4.1: Glossary of Z notation used in this thesis

| **Notation** | **Definition** | **Notation** | **Definition** |
|---|---|---|---|
| $p \Leftrightarrow q$ | Logical equivalence | $p \Rightarrow q$ | Logical implication |
| $\forall X \bullet q$ | Universal quantification | $\exists X \bullet q$ | Existential quantification |
| $\{ x, y, \ldots \}$ | Set display | $s \frown t$ | Sequence concatenation |
| $i_1, i_2, \ldots, i_n : a$ | Instances of type | $\mathbb{N}_1$ | Set of positive natural numbers |
| $A \times B \ldots$ | Cartesian product | $\# A$ | Number of elements in a set |
| $\mathbb{R}$ | Real number | $\mathsf{div}$ | Division |
| $\mathbb{P}$ | Power set | $f(x)$ | Function application |
| $A \rightarrow B$ | Total function | $A \nrightarrow B$ | Partial function |
| $::=$ | Definition | $\mathbb{R}^{m \times n}$ | Matrix |
| $(e_1, e_2)$ | Pair | $max\ A$ | Maximum of a set |
| $(x, y, \ldots)$ | Tuple | $i \mathrel{..} j$ | Number range |
| $seq\ A$ | Set of infinite sequences | $\langle x, y, \ldots \rangle$ | Sequence display |
| $\approx$ | Approximate equality | $s \mathsf{\ in\ } t$ | Sequence/tuple segment relation |
| $p \wedge q$ | Logical conjunction | $second\ x$ | Second element of pair |

Table 4.2: Types and Relations

| **Notation** | **Definition** | **Notation** | **Definition** |
|---|---|---|---|
| $U$ | Set of users | $R$ | Set of roles |
| $L$ | Set of locations | $T$ | Set of time |
| $S$ | Set of systems services | E | Set of events |
| $ITS$ | Set of itemsets | $FITS$ | Frequent itemset |
| $MAG$ | Maximal association group | $CEC$ | Constrained event cluster |
| $USC$ | User sequence cluster | $TCUSC$ | Time-constrained user sequence cluster |
| $SBSP$ | Subsequence patterns | $FSBSP$ | Frequent subsequence pattern |
| $RS$ | Representative sequence | $FSBSPC$ | Frequent subsequence pattern cluster |
| $B$ | Behavior pattern | $ItemC$ | Item constraint expression |
| $TimeC$ | Time constraint expression | $SizeC$ | Size constraint expression |
| $minsup$ | Minimum support threshold | $Dur$ | User sequence duration |
| $covT$ | Coverage threshold | $rep$ | Mapping of representative |

Table 4.1 lists the Z notations that are utilized in this thesis, and Table 4.2 lists a number of types, functions and relations that are introduced to formalize the behavior pattern mining model. These types and relations are employed to express user behavior pattern mining system as follows:

**Encoding&Attribute-Analysis**

- $U$ is a set of users where a user $u_i \in U$ is a person.
  $U ::= \{u_1, u_2, ..., u_i, ...\}$
- $R$ is a set of roles where a role $r_i \in R$ defines the user's responsibility within an organization.
  $R ::= \{r_1, r_2, ..., r_i, ...\}$
- $L$ is a set of locations where a lotion $l_i \in L$ indicates the place of the user issuing an access request.
  $L ::= \{l_1, l_2, ..., l_i, ...\}$
- $T$ is a set of times where a time instance $t_i \in T$ indicates the time stamp when the event was triggered. A time instance may be hours, minutes or seconds, depending on the required time accuracy.
  $T ::= \{t_1, t_2, ..., t_i, ...\}$
- $S$ is a set of services where a service $s_i \in S$ indicates the user issured service. More attributes may be added if the target system records more information in audit logs. For example, healthcare systems may record the emerging situation, and delegation rules or patient consent for privacy and security requirements [15].
  $S ::= \{s_1, s_2, ..., s_i, ...\}$
- $E$ is a set of events where each event $e_i$ is a tuple of attribute values.
  $E ::= \{e_1, e_2, ..., e_i, ...\}$
  $e_i = (a_{1i}, a_{2i}, a_{3i}, a_{4i}, a_{5i})$, where $a_{1i} \in U$, $a_{2i} \in R$, $a_{3i} \in T$, $a_{4i} \in L$, $a_{5i} \in S$
  $E \subseteq U \times R \times T \times L \times S$
- *ITS* is a set of itemsets where an itemset *its* is a tuple of attribute values with different size occurring in event database. For example, $(u_1, l_3)$ and $(u_2, t_5, s_3)$ are length-2 itemset and length-3 itemset respectively. An event $e$ contains an itemset *its* if $e$ contains every attribute value in *its*.

$$
\begin{array}{|l}
its : ITS \\
\hline
\exists\, events : \mathbb{P}\, E \bullet \forall\, event : events \bullet \\
\quad its \ \mathsf{in}\ event
\end{array}
$$

- *FITS* is a set of itemsets occurring frequently in event database $E$. The following schema defines an operation *FrequentItemsetMining*. *minsup*? is the input of the operation which ends in a question mark by convention; *FITS* is the output of the operation which ends in a exclamation mark. A frequent itemset *fits* is an itemset appearing in at leaset *minsup*? events from the event database. *sup* is a function mapping itemset to the number of times the itemset appears in the event database. Divided by the total number of events, the scope of itemset support is normalized to a real number between 0 and 1. *FITS*! is the output of frequent itemset mining on event database $E$ by given an input parameter *minsup*?.

$$
\begin{array}{|l}
\hline
\textit{FrequentItemsetMining} \\
minsup? : \mathbb{R} \\
FITS! : \mathbb{P}(ITS) \\
sup : ITS \rightarrow \mathbb{R} \\
\hline
0 < minsup? < 1 \\
sup(its) = \#\{e : E \mid its \ \mathsf{in}\ e\}\ div\ \#E \\
\forall\, fits : FITS! \bullet sup(fits) \geq minsup? \\
\hline
\end{array}
$$

- *MAG* is a set of maximal association groups, where a maximal association group $mag \in MAG$ defines maximal association in a group of events in the form of a maximal set of events that all share a maximum number of attribute values. The group of events is denoted by container *cnt* and the shared attribute values is denoted by $fits \in FITS$.

$$\begin{array}{|l}
mag : MAG \\
fits : FITS \\
cnt : \mathbb{P}(E) \\
\hline
mag = (fits, cnt) \\
\forall\, e : cnt \bullet fits \ \mathsf{in}\ e
\end{array}$$

<u>**Constrained Event Cluster**</u>

- *CEC* is a constrained event cluster where all events inside the cluster must satisfy data analysts specified constraints. The constraint reflects the common context of behaviors discovered from a *CEC*. The data analyst may produce a set of *CEC* for further behavior analysis respectively. The following schema defines an operation *ConstrainedCluster*. The input of the operation are: *source*? indicates the events of constrained cluster are from one or more maximal association groups; size constraint *SizeC*? limits the cluster size; and item constraint *ItemC*? enforces the filtering on events that are assigned to *CEC* by attribute values. The output *CEC*! is a set of events. The operation *ConstrainedCluster* selects events from search domain *source*? based on maximal association and assigns the ones that contain attributes denoted in item constraint *ItemC* into *CEC!*; and the size of selected events is controlled by size constraint *SizeC*. Maximal association measure is introduced in Section 4.2, and cluster constraints are explained in Section 3.4.

*ConstrainedCluster*

$source? : \mathbb{P}\{e : E \mid e \in second\ mag\}$

$SizeC? : \mathbb{N}_1$

$ItemC? : \mathbb{P}(U \cup R \cup L \cup T \cup S)$

$CEC! : \mathbb{P}\,E$

---

$\#CEC! \approx SizeC?$

$\forall\, e : CEC! \bullet$

$\quad e \ \mathsf{in}\ source? \wedge ItemC\ \mathsf{in}\ e$

- *USC* is a user sequence cluster that is obtained from constrained event cluster *CEC*. Each user sequence $us \in USC$ is a list of ordered events performed by the same user.

$USC : \mathbb{P}(\mathrm{seq}\, CEC)$

---

$\forall\, us : USC \bullet \exists\, u : U \bullet$

$\quad e \ \mathsf{in}\ us \wedge u\ \mathsf{in}\ e$

- *TCUSC* is a cluster of time constrained user sequences. Without time constraint, the length of user sequence $us \in USC$ is exploding as the timeline may be months even years. The length of user sequence is controlled by introducing the time constrain *TimeC* such as a minute, a day, or a week. A schema of producing *TimeConstrainedUserSequences* is defined, with input variable of time constraint $tc?$, and output variable $TCUSC!$. After performing the operation *TimeConstrainedUserSequences*, a user sequence $us \in USC$ becomes a number of limited ordered list of events performed by the same user within specified time interval. The function *Dur* indicates the duration of a sequence.

$TimeC ::= \{minute, hour, day, week, month\}$

*TimeConstrainedUserSequences*

$tc? : TimeC$

$TCUSC! : \mathbb{P}(seqCEC)$

$Dur : TCUSC! \rightarrow \mathbb{N}_1$

---

$\forall\, us : USC \bullet \exists\, s_1, s_2, ...s_n : TCUSC! \bullet$

$\quad us = s_1 \frown s_2 \frown ... \frown s_n$

$\forall\, tcus : TCUSC! \bullet \exists\, us : USC \bullet$

$\quad tcus \text{ in } us \wedge Dur(tcus) \leq tc?$

**User Sequence Analysis**

- *SBSP* is a set of subsequence patterns where a subsequence pattern *sbsp* is the subsequence of a group of sequences in *TCUSC*. The length of *sbsp* is determined by the number of sequence items. Each item of *sbsp* is either an individual attribute value or an itemset (a set of attribute values). For example, $(s_1, s_3, s_4)$ and $((u_2, l_5), (u_2, l_6))$ are length-3 subsequence and length-2 subsequence respectively.

$SBSP : \mathbb{P}(\text{seq}\, ITS)$

---

$\forall\, sbsp : SBSP \bullet \exists\, seqs : \mathbb{P}\, TCUSC \bullet \forall\, sequence : seqs \bullet$

$\quad sbsp \text{ in } sequence$

- *FSBSP* is a set of subsequence patterns occurring frequently in *TCUSC*. A frequent subsequence pattern is a subsequence appearing in at least *minsup* user sequences from *TCUSC*, where *minsup* is a parameter given by data analysts. *sup* is a function mapping a subsequence to how many times the subsequence appears in *TCUSC*. *FSBSP* is the output of frequent subsequence pattern mining on *TCUSC* by given an input parameter *minsup*?.

*FrequentSubsequencePatternMining*

$minsup? : \mathbb{R}$

$FSBSP! : \mathbb{P}(SBSP)$

$sup : SBSP \rightarrow \mathbb{R}$

---

$0 < minsup? < 1$

$sup(sbsp) = \#\{tcus : TCUSC \mid sbsp \text{ in } tcus\}\ div\ \#TCUSC$

$\forall fsbsp : FSBSP! \bullet sup(fsbsp) \geq minsup?$

- Similarity-based clustering algorithm is performed on *FSBSP* to group similar frequent subsequence patterns. The closeness of two sequences is represented by the maximum number of identical items in these two sequences (preserving the symbol order), which is defined as Longest Common Subsequence (LCS)[41] of the sequences. An extended LCS computing is introduced in Section 4.3 to evaluate the closeness among frequent subsequences. When the number of clusters is fixed to $n$, a function *sim* is defined to measure the similarity between sequences (assume $m$ is the number of frequent subsequence patterns). The clustering algorithm divides *FSBSP* into a set of frequent subsequence pattern clusters $FSBSPC = \{FSBSPC_1, FSBSPC_2, ..., FSBSPC_n\}$.

*FrequentSubsequencePatternClustering*

$$n? : \mathbb{N}_1$$

$$sim : FSBSP \times FSBSP \rightarrow \mathbb{R}^{m \times m}$$

$$clustering : FSBSP \rightarrow 1..n$$

---

$$\forall fsbsp : FSBSP \bullet clustering(fsbsp) = 1..n$$

---

- Representative sequences $RS$ is a small set of non redundant representatives covering a desired percentage of all subsequence patterns in a frequent subsequence pattern cluster $FSBSPC_i$. We use the same extended LCS (explained in Section 4.3) to measure similarity between sequences (assume $m$ is the number of frequent subsubsequence patterns in cluster $FSBSPC_i$). $rep$ is a partial function from $FSBSPC_i$ to $RS!$ which represents the mapping of a frequent subsequence pattern to its representative. *Density* is selected as representative criterion, which means an instance is more likely to be a candidate representative if it has more neighbourhoods in a dense region. $covT$ is coverage threshold that indicates the proportion of frequent subsequence patterns in $FSBSPC_i$ should have a representative in $RS!$.

*RepresentativeExtraction*

$$sim? : FSBSPC_i \times FSBSPC_i \rightarrow \mathbb{R}^{m \times m}$$

$$covT? : \mathbb{R}$$

$$RS! : \mathbb{P}(FSBSPC_i)$$

$$rep : FSBSPC_i \nrightarrow RS!$$

---

$$0 \leq covT? \leq 1$$

$$\#\{fsbsp : FSBSPC_i \mid rep(fsbsp) \in RS!\} \ \mathsf{div} \ \#FSPC_i \geq covT$$

---

**User Common Behavior Pattern**

- $B$ is a set of user behavior patterns finally discovered. $B_i$ represents a behavior pattern as a quadruple: $B_i a$ denotes the actor of behavior, which is extracted from cluster item constraints $ItemC$ about attributes user $U$ and role $R$; $B_i c$ denotes the context, which is extracted from constraints $ItemC$ about non-user and non-role attributes; $B_i s$ denotes the sequence, which is extracted from representative sequences $rs \in RS$; $B_i t$ denotes the time constraint, which is extracted from sequence time constraint $TimeC$.
  $B ::= \{B_1, B_2, ..., B_i, ...\}$

$$
\begin{array}{|l}
ItemC : CEC \\
TimeC : TCUSC \\
rs : RS \\
\hline
\exists\, B_i : B \bullet B_i = (B_i a, B_i c, B_i s, B_i t) \\
B_i a = (ItemC \cap U) \cup (ItemC \cap R) \\
B_i c = (ItemC \cap L) \cup (ItemC \cap T) \cup (ItemC \cap S) \\
B_i s = rs \\
B_i t = TimeC
\end{array}
$$

## 4.2 Computing Maximal Association

Without prior knowledge, we apply association mining on the encoded event database for discovering knowledge such as the most associated events and the interesting attributes. It is intended to bring together highly related events, and the shared attributes that indicate the common context of the group of events. *Maximal association* can be extracted by data mining and is considered as an interesting property for visualizing the structure of relations among groups of entities.

In this thesis, we use the notion of maximal association to define two similarity measures: similarity measure between events inside a mag (Maximal Association Group); and the similarity measure between two mags.

### 4.2.1 Association Measure of MAG

On the assumption that an event is more significant if it is associated with a large number of events through sharing more common attributes, such a association measure is used by data analysts in selecting events for subsequent behavior pattern mining. In this thesis, we also assume attributes have different significance, such as role contributes more than service in most systems. Intuitively, more important attributes will be assigned higher weights. An attribute weight setting algorithm may be provided by data analysts through a empirical analysis of the real world system. The events in a group are more associated if a large number of events in the group share more significant attribute values. The association-based similarity measure between events in maximal association group $mag_i$, denoted as $assoc(mag_i)$, is defined as follows:

$$assoc(mag_i) = \sum_{a \in fits} w_a + w_c * \log \mid cnt \mid$$

where $0 \leq w_a \leq 1$ is the weight of each shared attribute $a \in fits$; $0 \leq w_c \leq 1$ is the weight of the container of events *cnt* compared with the shared attributes. The association measure encodes both the size and the structure of $mag_i$. In general, the number of shared attributes contributes more on the closeness of events than the number of sharing events, if a group of events are considered for their similarity. Besides, the number of sharing events is much more than the number of shared attributes so that we use *logarithm* to make a balance between the values. Following formula normalizes the association value to the range between 0 and 1 by dividing the maximum association value of *MAG*.

$$assoc_f(mag_i) = \frac{assoc(mag_i)}{max_{MAG}\, assoc}$$

### 4.2.2 Association Measure between two MAGs

Each mag represents highly similar events and is considered as the building block for the proposed behavior pattern mining approach. In general, the association mining will produce a large number of mags. Few behavior patterns can be extracted from small mag. To address this issue. this thesis proposes an association measure between two mags. After examining the association between mags, data analysts have an opportunity to merge highly associated small mags to form a larger cluster. We consider two mags are similar if they have more common events sharing the same set of attributes. The association-based similarity measure between two maximal association groups $mag_i$ and $mag_j$, denoted as $assoc(mag_i, mag_j)$, is defined as follows:

$$assoc(mag_i, mag_j) = \textstyle\sum_{a \in fits_i \wedge a \in fits_j} w_a + w_c * \log \mid cnt_i \cap cnt_j \mid$$

where $0 \leq w_a \leq 1$ is the weight of each shared attribute among events in two mags $a \in fits_i \wedge a \in fits_j$; $0 \leq w_c \leq 1$ is the weight of the overlap of container events $cnt_i \cap cnt_j$ compared with the shared attributes. Following formula normalizes the association value to the range between 0 and 1 by dividing the maximum association value between maximal association groups *(MAG, MAG)*.

$$assoc_f(mag_i, mag_j) = \frac{assoc(mag_i, mag_j)}{max_{(MAG,MAG)}\, assoc}$$

## 4.3 Extended LCS

The length of LCS is considered as a measure of the closeness of two sequences, which finds the maximum number of identical items in these two sequences (preserving the event order). Each element of the sequence may be an itemset, but LCS algorithm can only compare simple items rather than itemset. For example, a sequence of behavior about actions (defined as attribute $A$) and accessed objects (defined as attribute $O$) looks like < *<A-1, O-1> <A-2, O-1> <A-3, O-2>* >. The itemsets *<A-1, O-1>* and *<A-1, O-2>* are partially identical. In this thesis, LCS (i.e., r(i, j)) is enhanced as follows, which allows comparing itemsets in sequence.

$$r(i,j) = \begin{cases} 0 & \text{if } i = 0 \vee j = 0 \\ r(i-1, j-1) + \dfrac{\mid lcs(x_i, y_i) \mid}{max\{\mid x_i \mid, \mid y_i \mid\}} & \text{if } \mid lcs(x_i, y_i) \mid > 0 \\ max\{r(i-1, j), r(i, j-1)\} & \text{if } \mid lcs(x_i, y_i) \mid = 0 \end{cases}$$

Let $X = (x_1, x_2, ..., x_m)$ and $Y = (y_1, y_2, ..., y_n)$ be two sequences of lengths $m$ and $n$, respectively. An element of the sequence, $x_i \in X$ and $y_j \in Y$, can be an itemset. Suppose the attribute values of an itemset ($x_i$ and $y_j$) are ordered, such as all elements in sequence $X$ and $Y$ follows the order of <Action, Resource, Location>. For example, $x_i =$ *<A-1, O-1, None>* and $y_j =$ *<A-1, None, L-2>*. The problem of comparing two itemset $x_i$ and $y_j$ can be converted to the problem of $lcs(x_i, y_j)$. A common subsequence cs of $x_i$ and $y_j$ represented by $cs(x_i, y_j)$ is a subsequence that occurs in both sequences. $lcs(x_i, y_j)$ is a common subsequence of both sequences with maximum length. The length of $lcs(x_i, y_i)$ is denoted by $\mid lcs(x_i, y_j) \mid$. Solving $\mid lcs(x_i, y_j) \mid$ is to determine the longest common subsequence for all possible prefix combinations of the two sequences $x_i$ and $y_i$. Let $r(i, j)$ be the length of the $lcs$ of $(x_1, x_2, ..., x_i)$ and $(y_1, y_2, ..., y_j)$. Then $LCS(X, Y)$ can be defined recursively using above formula.

## 4.4 Computational Complexity Overview

The complexity analysis is performed with respect to the formal specification of the algorithm in Section 4.1.

- **FrequentItemsetMining:** Apriori frequent itemset mining algorithm uses a bottom up approach, where frequent subsets are extended one item at a time, and groups of candidates are tested against the event database. There are two maim steps of Apriori algorithm: i) use frequent (k-1)-itemsets to generate candidates of frequent k-itemsets; and ii) scan event database and count each frequent k-itemsets candidate, and prune candidates if its support is below *minsup*. The bottleneck of Apriori algorithm is candidate generation. In the worst case, $10^4$ frequent 1-itemset will generate $10^7$ candidate 2-itemsets. Candidate test incurs k-times scan of event database. However, the average running time of Apriori algorithm is promising if given a reasonable *minsup* as lots of candidates are pruned in each round.

- **ConstrainedCluster:** K-mean clustering algorithm complexity is *O(nkdi)*, where n is the number of d-dimensional vectors, k the number of clusters and i the number of iteration needed until convergence. In our algorithm, we defined association-based similarity and the matrix has been built based on the extracted MAGs; n is the number of events from selected MAGs which is normally much smaller than the size of event database.

- **TimeConstrainedUserSequences:** Scan the constrained cluster once to split the user sequence into time constrained sequences.

- **FrequentSubsequencePatternMining:** The algorithm of frequent subsequence pattern mining is similar to Apriori frequent itemset mining, where the bottleneck is still candidate generation which becomes even worse. The maximum pattern size k of frequent itemset is the number of attributes defined in an event. In our case defined in 4.1, the maximum value of k is 5. As for sequence patterns, the pattern length is infinite for temporal data. For example, to discover a frequent subsequence pattern of length 100, the algorithm needs to generate $2^{100} \approx 10^{30}$ candidates. Thus we apply time constrains to user sequences before applying this algorithm to limit the maximum length of discovered sequence patterns.

- **FrequentSubsequencePatternClustering:** We applied hierarchical clustering algorithm on frequent subsequence patterns (size is m), where computational complexity is $O(m^2 \log m)$. As for computing the similarity between two sequences, the time complexity of *O(pt)* is required, where p and t are the length of sequences respectively.

- **RepresentativeExtraction:** We applied neighborhood density as the criteria to measure the representative score of frequent subsequence pattern. The algorithm of extracting most representative sequences requires computational complexity of $O(l^2)$, where l is the number of frequent subsequence patterns in a sequence cluster.

# Chapter 5

# Synthesizing Behavior-based Dataset

There is a general lack of access to real-world audit logs, and in particular in sensitive and mission-critical industries, such as healthcare, banking, and military. Moreover, finding or constructing a useful dataset from real-world systems is difficult due to the nondeterministic nature of the real-datasets especially in the early stages of the system during the production operation [57]. In general, the performance of different data mining approaches depend heavily on the testing dataset. Some algorithms are very sensitive to the target patterns and critical features. To decrease the algorithm evaluation time at the early stages, the data analysts can take advantage of a controllable simulation environment which enables acquiring various simulated dataset containing embedded interesting patterns and features. This chapter presents an interactive data exploration environment consisting of a design-generate-analyze-optimize process which assists data analysts in designing behavior-based dataset with embedded behavior patterns. A prototype toolkit named *"EventGenerator"* has been developed to synthesize dataset in different application domains.

This chapter is organized as follows. Section 5.1 introduces a behavior-based dataset generator toolkit for controllable dataset generation. Section 5.2 presents the dataset design using attribute distribution and behavior pattern definition. Section 5.3 explains the algorithm of behavior-based dataset generation. Section 5.4 provides two case studies to prove that the dataset generator toolkit is useful for generating dataset in different application domains.

## 5.1 Introduction

Most of the existing dataset generators create sequential dataset or clustering dataset with no specific goal. IBM Quest Synthetic Generator [58] is an open source market-basket synthetic dataset generator, capable of creating sequence dataset with randomly injected sequential patterns based on user specified parameters: sequence count, sequence pattern count, average sequence pattern length, sequence correlation, etc. The generated dataset is only suitable to evaluate the performance of data mining algorithms because of the extremely limited control on injected sequential patterns. SPMF (Sequential Pattern Mining Framework) [59] is an open-source data mining library that offers implementation of 75 data mining algorithms for sequential pattern mining, association mining, frequent itemset mining, and clustering. SPMF also provides a sequence dataset generator by using completely random method. Another synthetic dataset generator for clustering and outlier analysis creates datasets based on different distribution and transformation [60]. Given the number of points and number of clusters, dataset could be generated according to the user specified distribution, density level, difficulty level and outlier level which are based solely on the mathematical parameters. This dataset generator targets for testing clustering and outlier analysis algorithms instead of user behavior analytics. All these tools are not capable of being used to evaluate the proposed approach as this thesis requires a controllable user-behavior-based dataset generator.

Creating datasets that represent real-world systems is challenging. A representative synthetic dataset of user-system interactions should exhibit realistic features such as: the frequency of user access requests that mimic both normal work days and busy system times; statistic biases towards certain type of actions (e.g., more reads than writes); highly associated features (e.g., specific users normally work at specific locations); and more importantly, the sequence that reflects the continuous or connected series of actions (e.g., what a user does next is impacted by what he or she did in the past few steps). To design a controllable dataset generator with realistic features, we analyzed the scenarios in several public and private datasets in the fields of healthcare [51, 61], banking [62] and traffic collision [63]. Based on the analysis results, we extract sequencing, timing and association rules to define a behavior pattern: sequencing requires that a series of steps occur in a certain order; timing limits the occurrence frequency of certain values; and association enforces that the occurrence of one value should result in one or more other values.

We developed a dataset generator toolkit *EventGenerator* for controllable dataset generation, suitable for unbiased evaluation of user behavior pattern mining algorithms. *EventGenerator* has three layers: i) behavior pattern representation layer; ii) dataset generation layer; and iii) dataset visualization and analysis layer. The behavior pattern representation layer defines a scenario as a behavior pattern based on sequencing, timing and association rules. This representation layer allows data analysts to design interesting features and patterns that will be injected into the dataset. The dataset generation layer creates dataset that are controlled by data size, data distribution, and the designed behavior patterns. The visualization and analysis layer provides an interactive exploration environment for visual analysis of the quality of generated dataset and a means to revise and enhance the generated dataset. Without such a generator it is impossible to test, evaluate and calibrate the algorithms that explore a system's production dataset, as the nature

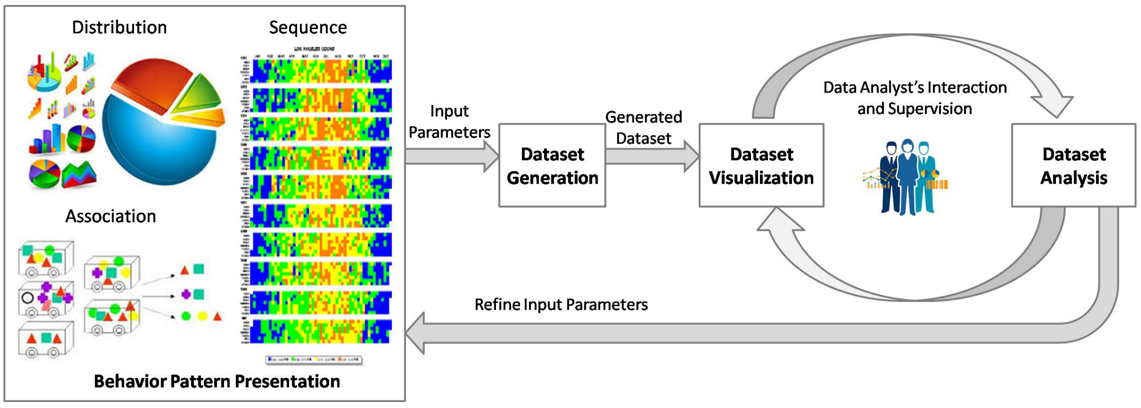


Figure 5.1: Synthetic dataset design, generation and visual analysis process

of a production dataset is unknown, and most probably it is not even available.

Figure 5.1 illustrates synthetic dataset design, generation and visual analysis process. The first important task is how to easily define the expected dataset. We propose a behavior pattern representation using data mining concepts (association pattern, and sequence pattern) as explained in Section 3.2 to constitute the behavior pattern, and mathematical tools (probability distribution) to indicate the statistical characteristics of data. The generation process produces a dataset that contains predefined attributes (representing the intended behavior patterns) and ensures that such behavior patterns will be embedded into the generated dataset. After producing the dataset according to the input parameters, the visual analysis step allows the data analysts to explore and verify the injected behavior patterns. The primary objective of the visualization is to extract simplified workable information from the dataset to effectively summarize and identify the embedded behavior patterns. How data analysts apply distinct data mining techniques (e.g., association mining, sequential pattern mining, and clustering) to visually analyse dataset is explained in Section 6.3. Using an iterative process, data analysts investigate the properties of the recovered behavior patterns, and finally refine the input parameters to optimize the generation process.

## 5.2 Dataset Design

Dataset generation process is controlled by: attribute distribution, and behavior pattern definition. Various continuous probability distributions (e.g., normal distribution, Poison distribution, beta distribution) can be applied to indicate the desired distribution of attribute values. We select normal distribution to explain our method because it is remarkably useful with a well-defined mean value and well-defined variance. We also propose a new abstraction of behavior pattern to assist data analysts in designing a desirable dataset.

### 5.2.1 Acquiring Valid Source Information

A key issue in using *"EventGenerator"* to produce high quality dataset is acquisition of valid source information about the key characteristics and user behaviors of real-world applications. In sensitive and mission-critical industries, the dataset designer analyzes audit log service specification to obtain information recorded in events, individual user-system interactions, and the sequence of user-system interactions to perform specific tasks in a scenario. Based on the analysis result, the dataset designer defines attributes, attribute distribution, behavior patterns and desired event size to control the produced dataset. In some cases, the real-world dataset is accessible, but the designer requires larger and more complicated dataset to evaluate the approach. The designer may analyze the existing small dataset to mine simple sequences of user-system interactions, and then merge simple sequences into user behavior patterns to simulate complicated scenarios.

### 5.2.2 Normal Distribution

Normal distribution [64] is specified by two parameters: the median value $\mu$ and standard deviation $\sigma$. The probability density function of random variable $x$ (the attribute value) with a normal distribution is as follows:

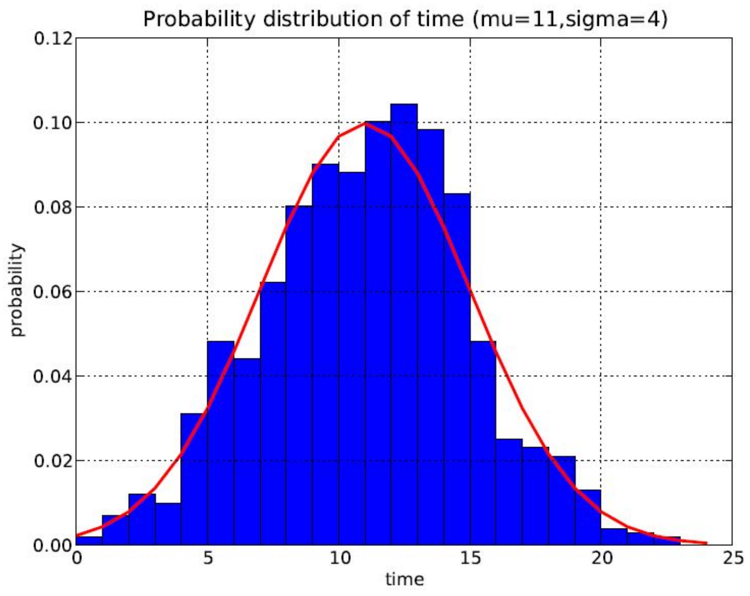


Figure 5.2: Normal distribution of attribute time with mean=11 and deviation=4

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\dfrac{(x-\mu)^2}{2\sigma^2}}$$

*EventGenerator* produces events with randomly selected attribute values but following specified distribution. By given an attribute name, attribute domain (the scope of allowed attribute values), mean value and standard variety value, data analysts are able to define desired attribute distribution in generated events. Figure 5.2 indicates an example of attribute distribution which simulates the time of accessing patient's health records in a hospital. A real system may have the following features: the hospital has to access patient's records the whole day (24 hours) as patient may come to hospital anytime; most of access requests are from 7:00 to 20:00; and the daily rush hour of system access is 11:00am. To simulate a dataset with such features, the data analysts can define an attribute named "time" with 24 hours; the mean value is 11:00am; and the deviation is 4 to simulate that daytime covers the most of access requests. Figure 5.2 shows the extracted attribute distribution of "time" from the generated dataset using *EventGenerator*.

### 5.2.3 Behavior Pattern Representation

A typical user behavior pattern can be expressed using behavior pattern representation "*Behavior = <Actor, Sequence, Context, Time-Interval, Support>*", where *Actor* is a specific user or a group of users; *Context* is explained by association to enforce that the occurrence of one attribute value should result in one or more other attribute values; *Sequence* requires that attribute values occur in a certain sequence; *Time-Interval* restricts the duration of behavior such as hourly behavior, daily behavior, or weekly behavior; and *Support* indicates the percentage of generated events should constitute this behavior pattern.

Following shows an example representation of user behavior patterns in healthcare system. The first behavior pattern *P-00001* describes a typical doctor's workflow in radiology department, of taking a medical exam for a patient in the following order: order an exam; read patient's historical exams; create a new exam for the patient; write diagnostic report for the new exam. This medical examination workflow should finish within 3 working days. The support specifies the percentage of events in which this behavior pattern exists. The second behavior pattern *P-00002* indicates daily nursing ward-round in hospital. In this example, nurses normally perform ward-round in order: *ward-A, ward-B, ward-C*. The support means around 15% of generated events should include behavior pattern *P-00002*.

In the example, the behavior pattern is expressed in JSON (JavaScript Object Notation) format [65]. JSON is built on two types of structures that fit our requirements very well: i) a collection of name-value pairs; and ii) an ordered list of values. The data analysts design and express various behavior patterns in this simple JSON format. The dataset generator would automatically transform the behavior definition to association patterns, sequence patterns and time constraints. These patterns and constraints control the dataset generation process.

```
[
  {
    "id": "P-00001",
    "description": "doctor's workflow in radiology
       department",
    "actor": [
      {"role": "doctor"}
    ],
    "context": {
      "department": "radiology"
    },
    "sequence": [
      {
        "action": "create an order"
      },
      {
        "action": "read historical exam"
      },
      {
        "action": "create an exam"
      },
      {
        "action": "create a report"
      }
    ],
    "duration": "3 days",
    "support": "10%",
  },
  {
    "id": "P-00002",
    "description": "daily nursing ward-round",
    "actor": [
      {"role": "nurse"}
    ],
    "sequence": [
      {
        "location": "ward-A"
      },
      {
        "location": "ward-B"
      },
      {
        "location": "ward-C"
      }
    ],
    "duration": "daily",
    "support": "15%"
  }
]
```

## 5.3 Dataset Generation

In this section, we formally define the proposed attribute distributions and behavior pattern definition. Let $A = \{a_1, a_2, ..., a_m\}$ be a set of attributes designed by the data analyst. Let $V = \{V_1, V_2, ..., V_m\}$ denotes the collection of the allowed domains of values for different attributes, where $V_i = \{v_{i1}, v_{i2}, ..., v_{ix}, ..., v_{in}\}$ denotes the attribute domain of attribute $a_i$, and $v_{ix}$ denotes a specific value of attribute $a_i$. Let $B = \{B_1, B_2, ..., B_j, ...\}$ be a set of behavior patterns designed by the data analyst. Each behavior pattern is represented as a tuple $B_j =< B_j a, B_j s, B_j c, B_j t, B_j sup >$, where $B_j a$ is actor, $B_j s$ is sequence, $B_j c$ is context, $B_j t$ is time interval, and $B_j sup$ is support.

### 5.3.1 Dataset Generation Algorithm

Table 5.1 lists the variables that are used in dataset generation algorithm. Algorithm 2 generates dataset with three main steps: i) parse and convert behavior patterns into association patterns, sequential patterns and time constraints, and randomly select users to support these behavior patterns; ii) build biased random attribute value selection function; and iii) generate event dataset according to the rules constructed in steps i) and ii). First, the Algorithm loads the data analyst's input parameters including the attribute definition $A$, attribute domain $V$, user behavior pattern design $B$, all system users $U_{All}$, and the average event number per user per day $avg$ and all event days $D_{All}$ to control the dataset size. The description of the Algorithm with reference to its line numbers is as follows:

Table 5.1: Variables used in dataset generator algorithm

| **Variables** | **Description** |
|---|---|
| $E = \{e_1, e_2, ..., e_i, ...\}$<br>$e_i =< v_{i1}, v_{i2}, ..., v_{im} >$<br>$u_o\text{-}d_p\text{-}s_x = <e_{op1},\ e_{op2},\ ...,\ e_{opx}>$ | $E$ is a set of events.<br>An event $e_i$ is a tuple of attribute values.<br>$u_o\text{-}d_p\text{-}s_x$ denotes a daily event sequence $s_x$ (length is $x$) of user $u_o$ at day $d_p$. |
| $B = \{B_1, B_2, ..., B_j, ...\}$<br>$B_j =< B_j\,a, B_j\,s, B_j\,c, B_j\,t, B_j\,sup >$ | $B$ is a set of predefined behavior patterns.<br>$B_j\,a$ denotes the actor of behavior $B_j$; $B_j\,s$ denotes the sequence; $B_j\,c$ denotes the context; $B_j\,t$ denotes the time constraint; $B_j\,sup$ denotes the support. |
| $I = \{i_1, i_2, ..., i_j, ...\}$<br>$i_j =< v_{j1}, v_{j2}, ... >$ | $I$ denotes a set of association patterns.<br>Each association pattern $i_j$ enforces a group of attribute values (e.g., $v_{j1}$ and $v_{j2}$) to occur together within a group of events, where the behavior context constraint $B_j\,c$ determines $i_j$ and the selected events. |
| $S = \{s_1, s_2, ..., s_j, ...\}$<br>$s_j =< v_{j1}, v_{j2}, v_{j3}, ... >$ | $S$ denotes a set of sequence patterns.<br>Each sequence pattern $s_j$ enforces a group of attribute values (e.g, $v_{j1}$, $v_{j2}$ and $v_{j3}$) to occur in time order and be inserted into an event sequence $u_o\text{-}d_p\text{-}s_x$, where the behavior sequence constraint $B_j\,s$ determines $s_j$ and the selected event sequence. |
| $U_{All} = \{u_1, u_2, ..., u_n\}$<br>$U = \{U\text{-}B_1, U\text{-}B_2, ..., U\text{-}B_j, ...\}$<br>$U\text{-}B_j = \{u_{j1}, u_{j2}, ..., u_{jl}\}$ | $U_{All}$ is the set of all users in the system.<br>$U$ denotes the collection of users that are selected to insert different behavior patterns. Unselected users do not have behavior patterns.<br>$U\text{-}B_j$ denotes a group of selected users $\{u_{j1}, u_{j2}, ..., u_{jl}\}$ to have predefined behavior pattern $B_j$.<br>$l$ is the number of selected users, which is controlled by the support of the behavior pattern $B_j\,sup$. |
| $F_k$ | $F_k$ denotes a distribution function for generating random values for attribute $a_k$. |
| $avg$ | $avg$ denotes the average number of generated events per user per day. |
| $D_{All}$ | $D_{All}$ denotes the collection of days.<br>$avg$ and $D_{All}$ are used to control the dataset size. |

**Algorithm 2** Dataset Generator Algorithm

```
INPUT: $A, V, B, avg, D_{All}, U_{All}$
OUTPUT: $E$
1: $I = \phi, S = \phi, E = \phi, U = \phi$
2: for all $B_j$ in $B$ do
3:    association pattern $i_j = B_j c$
4:    sequence pattern $s_j = B_j s$
5:    apply time constraint $B_j t$ to $s_j$
6:    randomly select $B_j sup$ users for $U$-$B_j$
7: end for
8: for all $a_k$ in $A$ do
9:    build biased random value select function $F_k$
10: end for
11: for all $u_o$ in $U_{All}$ do
12:    for all $d_p$ in $D_{All}$ do
13:       $x$ = randomly selected an integer around $avg$
14:       generate an empty event sequence $u_o$-$d_p$-$s_x$
15:       for all $U$-$B_j$ in $U$ do
16:          if $u_o$ exists in $U$-$B_j$ then
17:             insert constrained sequence pattern $s_j$ into $u_o$-$d_p$-$s_x$
18:             insert association pattern $i_j$ into $u_o$-$d_p$-$s_x$
19:          end if
20:       end for
21:       for all event $e_i$ in $u_o$-$d_p$-$s_x$ do
22:          for all empty attribute $a_k$ in $e_i$ do
23:             call function $F_k$ to assign random value $v_{ik}$ to $e_i$
24:          end for
25:       end for
26:    end for
27: end for
```

- **Line 2 to 7:** transforms each behavior pattern into association patterns and constrained sequence patterns. As each behavior pattern is defined with a support parameter, the algorithm randomly selects $B_j sup$ users from the system users $U_{All}$ to support the behavior pattern $B_j$.

- **Line 8 to 10:** builds a set of biased random attribute value selection functions that follow the attribute distribution definition.

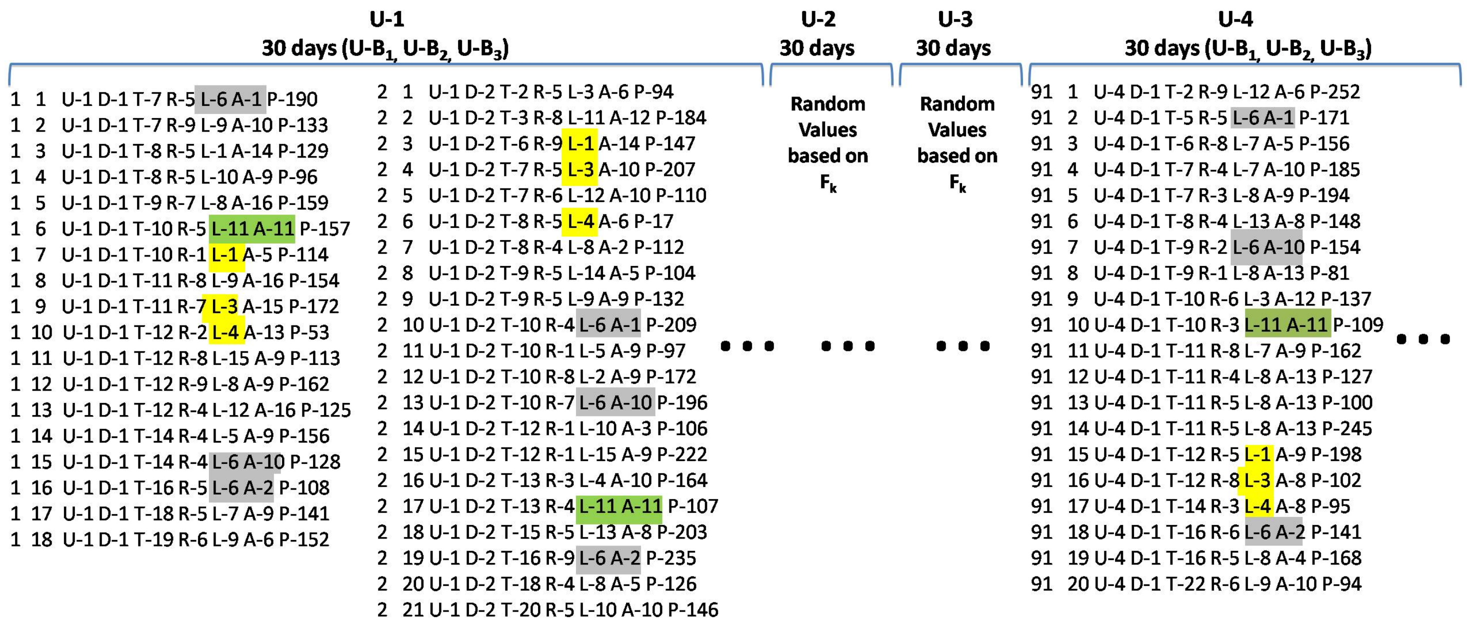
- **Line 11 to 14:** starts to generate the events user by user, day by day. Line-13 assigns a random integer $x$ around the value of $avg$ as the number of events for user $u_o$ at day $d_p$. Line-14 generates an empty event sequence $u_o$-$d_p$-$s_x$ for user $u_o$ at day $d_p$ with length of $x$.

- **Line 15 to 20:** if the user $u_o$ is selected to support one or more behavior patterns, the algorithm inserts the corresponding association patterns and constrained sequence patterns into the event sequence $u_o$-$d_p$-$s_x$. Based on the time-constraints, the Algorithm must restrict the duration of events when inserting sequence patterns into event sequence. After inserting the patterns, the event sequence is only partially generated since some attributes that are not defined in patterns are still empty.

- **Line 21 to 25:** calls the biased random attribute value selection function to assign values to the remaining attributes. If the user is not selected to insert any pattern, the algorithm executes section from line-21 to line-25 to randomly assign all attribute values to each event.

### 5.3.2 Dataset Generation Output

Figure 5.3 shows a slice of the generated events for users U-1, U-2, U-3 and U-4 for a month (30 days), using above dataset generator algorithm. The average events per user per day is 20; day D-1 of user U-1 has 18 events; day D-2 of user U-1 has 21 events; day D-1 of user U-4 has 20 events. Each row represents an event in the format of "*seuqnece_id event_id user date time role location action patient*". The first column is sequence id and the second column is event id within a sequence. Each attribute value is encoded in the format of a representation letter followed by an integer. For example, D-2 represents the second day of the month. Consider three predefined behavior patterns $B_1$, $B_2$ and $B_3$ such that $B_1s$ = {L-1, L-3, L-4}, $B_1t$ = {4 hours}, $B_2s$ = {A-1, A-10, A-2}, $B_2c$ = {L-6}, $B_3c$ = {A-11, L-11}. When performing the event generator Algorithm, consider users U-1 and U-4 are selected to contain these three behavior patterns (i.e., $U$-$B_1$ = {U-1, U-4}, $U$-$B_2$



U-1
30 days (U-B$_1$, U-B$_2$, U-B$_3$)

1 1 U-1 D-1 T-7 R-5 L-6 A-1 P-190
1 2 U-1 D-1 T-7 R-9 L-9 A-10 P-133
1 3 U-1 D-1 T-8 R-5 L-1 A-14 P-129
1 4 U-1 D-1 T-8 R-5 L-10 A-9 P-96
1 5 U-1 D-1 T-9 R-7 L-8 A-16 P-159
1 6 U-1 D-1 T-10 R-5 L-11 A-11 P-157
1 7 U-1 D-1 T-10 R-1 L-1 A-5 P-114
1 8 U-1 D-1 T-11 R-8 L-9 A-16 P-154
1 9 U-1 D-1 T-11 R-7 L-3 A-15 P-172
1 10 U-1 D-1 T-12 R-2 L-4 A-13 P-53
1 11 U-1 D-1 T-12 R-8 L-15 A-9 P-113
1 12 U-1 D-1 T-12 R-9 L-8 A-9 P-162
1 13 U-1 D-1 T-12 R-4 L-12 A-16 P-125
1 14 U-1 D-1 T-14 R-4 L-5 A-9 P-156
1 15 U-1 D-1 T-14 R-4 L-6 A-10 P-128
1 16 U-1 D-1 T-16 R-5 L-6 A-2 P-108
1 17 U-1 D-1 T-18 R-5 L-7 A-9 P-141
1 18 U-1 D-1 T-19 R-6 L-9 A-6 P-152

2 1 U-1 D-2 T-2 R-5 L-3 A-6 P-94
2 2 U-1 D-2 T-3 R-8 L-11 A-12 P-184
2 3 U-1 D-2 T-6 R-9 L-1 A-14 P-147
2 4 U-1 D-2 T-7 R-5 L-3 A-10 P-207
2 5 U-1 D-2 T-7 R-6 L-12 A-10 P-110
2 6 U-1 D-2 T-8 R-5 L-4 A-6 P-17
2 7 U-1 D-2 T-8 R-4 L-8 A-2 P-112
2 8 U-1 D-2 T-9 R-5 L-14 A-5 P-104
2 9 U-1 D-2 T-9 R-5 L-9 A-9 P-132
2 10 U-1 D-2 T-10 R-4 L-6 A-1 P-209
2 11 U-1 D-2 T-10 R-1 L-5 A-9 P-97
2 12 U-1 D-2 T-10 R-8 L-2 A-9 P-172
2 13 U-1 D-2 T-10 R-7 L-6 A-10 P-196
2 14 U-1 D-2 T-12 R-1 L-10 A-3 P-106
2 15 U-1 D-2 T-12 R-1 L-15 A-9 P-222
2 16 U-1 D-2 T-13 R-3 L-4 A-10 P-164
2 17 U-1 D-2 T-13 R-4 L-11 A-11 P-107
2 18 U-1 D-2 T-15 R-5 L-13 A-8 P-203
2 19 U-1 D-2 T-16 R-9 L-6 A-2 P-235
2 20 U-1 D-2 T-18 R-4 L-8 A-5 P-126
2 21 U-1 D-2 T-20 R-5 L-10 A-10 P-146

• • •

U-2
30 days

Random Values based on F$_k$

• • •

U-3
30 days

Random Values based on F$_k$

• • •

U-4
30 days (U-B$_1$, U-B$_2$, U-B$_3$)

91 1 U-4 D-1 T-2 R-9 L-12 A-6 P-252
91 2 U-4 D-1 T-5 R-5 L-6 A-1 P-171
91 3 U-4 D-1 T-6 R-8 L-7 A-5 P-156
91 4 U-4 D-1 T-7 R-4 L-7 A-10 P-185
91 5 U-4 D-1 T-7 R-3 L-8 A-9 P-194
91 6 U-4 D-1 T-8 R-4 L-13 A-8 P-148
91 7 U-4 D-1 T-9 R-2 L-6 A-10 P-154
91 8 U-4 D-1 T-9 R-1 L-8 A-13 P-81
91 9 U-4 D-1 T-10 R-6 L-3 A-12 P-137
91 10 U-4 D-1 T-10 R-3 L-11 A-11 P-109
91 11 U-4 D-1 T-11 R-8 L-7 A-9 P-162
91 12 U-4 D-1 T-11 R-4 L-8 A-13 P-127
91 13 U-4 D-1 T-11 R-5 L-8 A-13 P-100
91 14 U-4 D-1 T-11 R-5 L-8 A-13 P-245
91 15 U-4 D-1 T-12 R-5 L-1 A-9 P-198
91 16 U-4 D-1 T-12 R-8 L-3 A-8 P-102
91 17 U-4 D-1 T-14 R-3 L-4 A-8 P-95
91 18 U-4 D-1 T-16 R-6 L-6 A-2 P-141
91 19 U-4 D-1 T-16 R-5 L-8 A-4 P-168
91 20 U-4 D-1 T-22 R-6 L-9 A-10 P-94

• • •

Figure 5.3: A slice of generated events using dataset generator algorithm

= {U-1, U4}, $U\text{-}B_3$ = {U-1, U-4}). First, the Algorithm parses these 3 behavior patterns and converts them into: association patterns $i_1$ = {A-1, L-6}, $i_2$ = {A-10, L-6}, $i_3$ = {A-2, L-6}, $i_4$ = {A-11, L-11}; and constrained sequence pattern $s_1$ = {L-1, L-3, L-4} which happened within 4 hours and sequence pattern $s_2$ = {A-1, A-10, A-2}. Then the Algorithm generates the event sequences for users U-1, U-2, U-3 and U-4 day by day, respectively. It generates an empty event sequence for user U-1 at day D-1 with the length of 18. Then the Algorithm randomly inserts association patterns $i_1$, $i_2$, $i_3$, $i_4$ and sequence patterns $s_1$ and $s_2$ into the event sequence. As shown in Figure 5.3, the inserted behavior patterns are highlighted: $B_1$ is marked as yellow, $B_2$ as gray, and $B_3$ as green. After that, the Algorithm assigns values to the remaining empty attributes based on their attribute distributions. Following the same flow, the Algorithm generates event sequence for User U-1 at day D-2 with 21 events, and after 30 days it generates event sequences for user U-2, which is not selected to insert any behavior pattern so that all attribute values of events for U-2 are randomly selected by function $F_k$ based on the intended attribute distributions. The events for user U-3 are also randomly generated based on random attribute selection function $F_k$. After 30 days for user U-3, the Algorithm generates 20 events for user U-4 at day D-1 with embedded behavior patterns, as shown in Figure 5.3.

## 5.4 Case Studies

To evaluate the functionality and feasibility of *EventGenerator*, we produced a dataset to simulate user-system interactions in distributed medical imaging systems. The attribute characteristics and designed user behavior patterns are selected after analyzing the audit log specification in distributed PACS systems (Picturing Archiving and Communication System). We use this case study to demonstrate the dataset design, dataset generation, and analysis. Besides, we designed two more datasets to simulate banking services and traffic collisions to prove that our dataset generator toolkit is useful for generating datasets in different application domains.

### 5.4.1 Dataset Design and Generation

It is very challenging to access the production audit logs from healthcare organizations due to the patient's sensitive information, privacy issues, and ethics board's approval. Therefore, a synthetic dataset is required to assist researchers and developers for systematic testing, analyzing and evaluating different techniques and software solutions to be approved for healthcare domain. Distributed PACS systems follow RFC3881 "Security Audit and Access Account ability Message XML Data Definitions for Health Applications" [66]. This document defines the minimum set of attributes that need to be captured for security auditing in healthcare application systems. In our experiment, an event is composed of a group of attributes based on RFC3881 definition, as follows: Event=<User, Location, Action, Patient, Date, Time>, where User is the user identifier; Location is the current location of user; Action is the service that the user requested; Patient is the patient identifier indicating the owner of the requested resource; Date and Time record the timestamp.

Table 5.2: Attribute distribution definition to simulate audit logs from medical image system

| **Attribute** | **Representation** | **Domain** | **Type** | **Mu** | **Sigma** |
|---|---|---|---|---|---|
| User | U- | 100 | Random | - | - |
| Location | L- | 15 | Normal | 8 | 8 |
| Action | A- | 16 | Normal | 8 | 3 |
| Patient | P- | 300 | Normal | 150 | 50 |
| Date | D- | 30 | Normal | 15 | 5 |
| Time | T- | 24 | Normal | 11 | 4 |

Table 5.2 indicates the input parameters of the attribute distribution. Attribute value is encoded by an "identifying letter" plus an integer number (e.g.,

U-1, D-2, L-4). The attribute domain specifies the scope of the allowed attribute values. In our experiment, we simulated a system consisting of 100 users with different roles (physicians, lab specialists, etc.) and 300 patients. The timeline of the simulated events is one month. The attribute Time records the hour when the event occurs. Minute and second can be added if data analysts require more precise time-stamp. The attributes with normal distributions need to be configured with two parameters: mean (mu) and variance (sigma). For example, the attribute Time with normal distribution and "mu=11" means "11:00am" is the rush hour in real systems within the hospitals.

Table 5.3: Behavior pattern definition to simulate audit logs from medical imaging system

| **Pattern Category** | **Pattern Id** | **Sequence** | **Support** |
|---|---|---|---|
| Location Sequence | P-00001 | office-1-Juravinski-Hamilton, office-3-Juravinski-Hamilton, office-4-Juravinski-Hamilton | 30% |
| | P-00002 | office-3-Lakeridge-Oshawa, office-4-Lakeridge-Oshawa, office-5-Lakeridge-Oshawa | 25% |
| | P-00003 | office-2-McMaster-Hamilton, office-1-McMaster-Hamilton, office-3-McMaster-Hamilton | 20% |
| Action Sequence | P-00004 | read exam, read report, update report | 30% |
| | P-00005 | read exam, read order, create exam | 25% |
| | P-00006 | create profile, read profile, update profile | 20% |
| Time Sequence | P-00007 | 11:00, 12:00, 14:00 | 30% |
| | P-00008 | 10:00, 11:00, 12:00 | 25% |
| | P-00009 | 14:00, 15:00, 16:00 | 20% |

Table 5.3 shows 9 typical user behavior patterns that constitute ordering, timing, sequence and combination relationships among events. Pattern P-00001 expresses that a user in a single day works at three different locations in the following order: office-1-Juravinski-Hamilton, office-3-Juravinski-Hamilton, office-4-Juravinski-Hamilton. Support is the percentage of users whose event sequences include this location sequence pattern. Pattern P-00002 and P-00003 are two more examples of location sequence patterns with different supports. P-00004 shows a

typical physician workflow in radiology department in the following order: read a new exam; read its diagnostic report; update the diagnostic report. P-00005 and P-0006 are also two typical workflows in a radiology department with different supports. P-00007 to P-00009 indicate three time sequence patterns during which physicians have accessed the medical images.

Our dataset generator transforms these 9 user behavior patterns onto the JSON formats as the input parameters. By adding more parameters with average event count per user per day to control the size of the dataset, our generator engine generated 30,000 events with embedded designed features.

### 5.4.2 More Application Domains

Various application domains can benefit from our designed toolkit by generating datasets with domain specific features and behavior patterns. For example, a Consultant Service in a banking organization is proposed that is capable of matching the customer's attributes with those of previous customers under similar circumstance to provide the new customer with the best possible recommendations [62]. Our dataset generator can be used to design and generate the desired dataset to train the recommendation system. Based on the investigation of several services offered by a Banking organization, we designed the attribute-tuple for banking event as "<*Status, Occuption, TypeOfService, Age, AmountOfMoney, UseOfMoney, CreditHisotry, Degree*>".

Table 5.4: Behavior pattern definition to simulate dataset of banking services

| **Pattern Id** | **sequence** | **Time Constraint** | **Support** |
|---|---|---|---|
| P-00001 | BB, BBB, AA, AAA | 4 months | 10% |
| P-00002 | AAA, BB, CCC | 3 months | 5% |
| P-00003 | Good, Good, Excellent | 3 years | 15% |
| P-00004 | Fair, Good, Poor | 3 years | 10% |
| P-00005 | Travel, Travel, House, House | 4 years | 15% |
| P-00006 | Tuition, Tuition | 2 years | 5% |

Using our dataset generator interface, the system designer is able to define the distribution of different attributes of the banking event and the banking service scenarios to be injected into a training dataset. Table 5.4 indices 6 example behavior patterns in the banking service systems. Suppose that a CreditHisotry System for bank customers records the customer's credit ratings from excellent to poor, as: AAA, AA, A, BBB, BB, B, CCC, CC, C and D, respectively. Pattern P-00001 expresses that a pattern of credit rating is improving since the past 4 months. On the other hand, pattern P-00002 is a pattern of credit rating is deteriorating in the past 3 months. People's income AmountOfMoney is divided into ranges of Excellent, Good, Fair, and Poor. Pattern P-00003 and P-00004 are two example patterns of income changing in recent 3 years. Attribute UseOfMoney tracks customer's expenses. Pattern P-00005 reveals young people mostly spend money on travel when they are single, and then spend a large amount of money on purchasing a house after marriage. P-00006 indicates normally students bear the burden of tuition.

Table 5.5: Behavior pattern definition to simulate dataset of traffic collision

| **Pattern Id** | **sequence** | **Support** |
|---|---|---|
| P-00001 | Snowing, Climbing lane | 5% |
| P-00002 | Young, Sport, Injury | 10% |
| P-00003 | 8:00am, No injury | 20% |

Table 5.5 presents another example regarding the traffic safety recommendation, which requires a detailed testing dataset when collisions happen, such as: collision level, vehicle's attributes, driver's attributes, weather and location. After investigating several Canada's traffic collision reports, we designed the attribute-tuple for traffic collision event as <CollisionTime, CollisionLocation, Critical, VehiclesInvolved, VehileModel, Weather, RoadSurface, DriverAge, DriverGender>. Depending on our generator interface, the safety recommendation system designer can produce various testing datasets with designed collision features. Table 5.5 shows 3 example behavior patterns based on association pattern to describe traffic collisions. Pattern P-00001 reveals that traffic collision is easy to happen in

the climbing lane during snow fall. Pattern P-00002 explains more young people driving sport vehicles have injury in traffic collision because of over speed. And many traffic collisions happen at 8:00am but with few injuries as 8:00am is rush hour.

# Chapter 6

# Case Studies

This chapter presents three case studies to demonstrate the functionality, performance and accuracy of the user behavior pattern recovery technique in the thesis. A prototype toolkit has been implemented to assist the system administrators in analyzing audit logs from distributed systems. The experiment in this chapter has been organized to:

- Demonstrate the generality of the proposed approach by experimenting with systems in different domains such as distributed medical imaging system, and public cloud computing service.
- Evaluate the usefulness of the approach in terms of producing high-quality results with recommendations, efficiency and accuracy of behavior pattern mining.
- Demonstrate the user involvement and the incorporation of domain/system knowledge in the user behavior pattern recovery process.

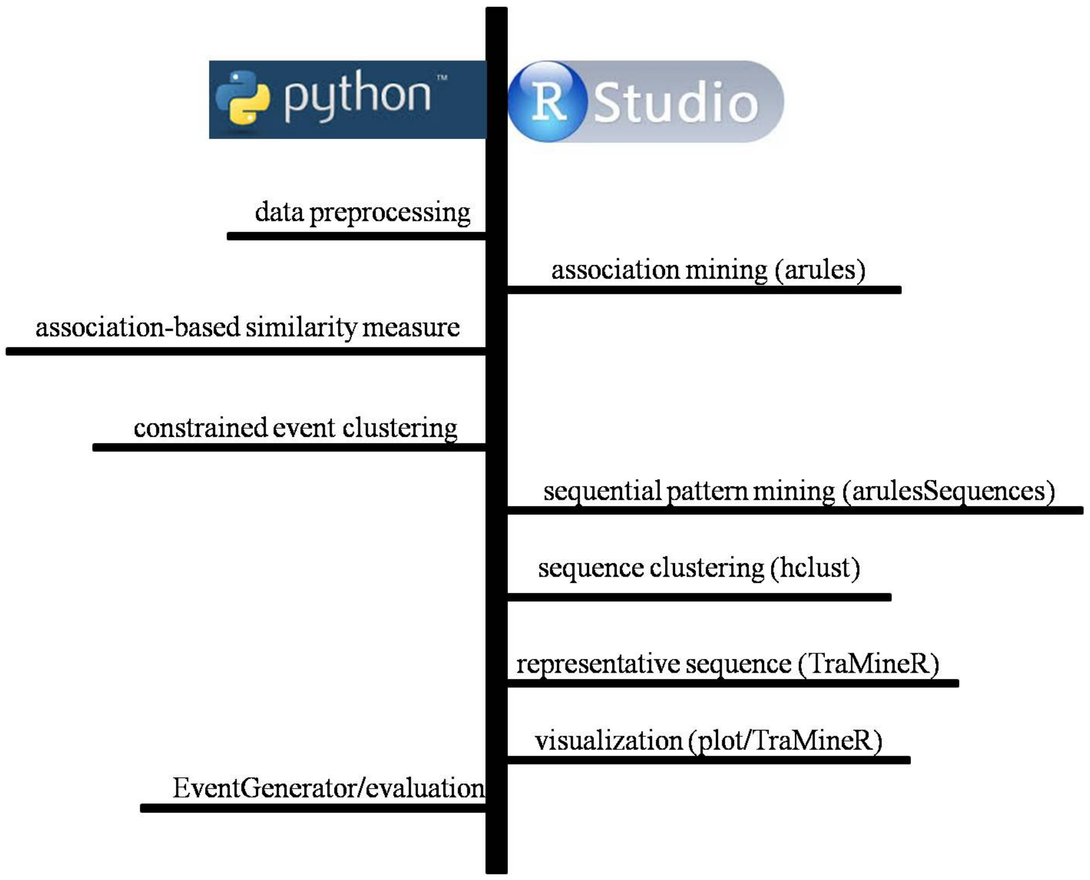

python
RStudio
data preprocessing
association mining (arules)
association-based similarity measure
constrained event clustering
sequential pattern mining (arulesSequences)
sequence clustering (hclust)
representative sequence (TraMineR)
visualization (plot/TraMineR)
EventGenerator/evaluation

Figure 6.1: Programming language, software and libraries used in experiment

## 6.1 Experimentation Platform and Tools

The hardware platform for the experiment consists of a Ubuntu 14.04 virtual machine on Oracle VirtualBox with 2.90GHz, 2GB memory, and 40G disk. As part of this work, the proposed user behavior pattern mining approach has been implemented in a toolkit. The toolkit offers an interactive environment for recovering and evaluating user behavior patterns, which provides the following functionalists: i) parsing and converting target system audit logs into encoded attributed events; ii) association-based similarity measure and constrained event clustering; iii) frequent sequential pattern mining as presented in [48]; iv) sequence clustering and representative extraction as presented in [49, 50]; and v) an evaluation of the recovered user behavior pattern using developed *EventGenerator*.

Figure 6.1 presents the proragmming languge, software and libraries used in experimentation. Our toolkit is developed using two progamming languges: python and R. "Python is a widely used high-level, general-purpose, interpreted, dynamic programming language, which lets you work quickly and integrate systems more effectively." [67] The collected audit logs from target systems are in JSON/XML format. Python provides standard libraries for parsing XML and JSON. Therefore, the audit log preprocessing and attributed event encoding are written in Python. R language is widely used in statistical computing, data mining and data analysis. Besides, many free R packages are supplied with collection of R functions, data structure and compiled code. Our developed toolkit about data analysis, pattern mining, clustering and visualization are built using R [45]. The packages that are used are also listed in figure 6.1.

*EventGenerator* is a useful utility which allows data analyst to produce customized events with user behavior patterns embedded. The data analyst can easily design user behavior patterns and generate events through configuration without having to write any code. *EventGenerator* is written in Python, which is deployed at GitHub [68]. The data analyst provides an attribute definition file, a user behavior pattern definition file, and a set of parameters to control the generated event size. *EventGenerator* can produce a 100K event dataset containing 10 attributes within 10 seconds. Each row of the output event file represents an event, with attribute values separated by space. At the same time, another file is generated which records where the patterns are inserted into. This file is used for evaluating the accuracy and completeness of our approach in subsection 6.4.2.

## 6.2 Distributed Medical Imaging System

We collected audit logs from distributed PACS (Picture Archiving and Communication System) and IHE (Integrating the Healthcare Enterprise) integrated imaging systems using MARC-HI Everest Framework [69]. This framework builds a higher level API that can be used directly by application developers for communication with remote systems in a standard manner. The source systems, PACS and IHE integrated imaging systems, follow the "RFC3881-Security Audit and Access Accountability Message XML Data Definitions for Healthcare Applications" [66] to generate audit logs. This document defines the format of data to be collected and the minimum set of attributes that need to be captured for security auditing in healthcare application systems.

Table 6.1: Attributes converted from audit log of distributed medical imaging system

| **Attribute Name** | **Data Definition in XML schema according to RFC3881** | **Attribute Domain** | **MinSup** |
|---|---|---|---|
| User | ActiveParticipant/UserIsRequestor='true'/UserID | 329 | 0.003 |
| Role | ActiveParticipant/RoleIDCode/code | 4 | 0.25 |
| Location | ActiveParticipant/ UserIsRequestor='true'/NetworkAccessPointId | 18 | 0.05 |
| Action | EventIdentification/EventActionCode | 3 | 0.3 |
| Paitent | ParticipantObjectIdentification/ ParticipantObjectTypeCode='1'/ParticipantObjectID | 84 | 0.01 |
| Resource | ParticipantObjectIdentification/ ParticipantObjectTypeCode='2'/ParticipantObjectID | 492 | 0.002 |
| Date | EventIdentification/EventDateTime | 25 | 0.04 |
| Time | EventIdentification/EventDateTime | 24 | 0.04 |

In the preprocessing stage, the collected audit logs are converted into attributed events according to the schema defined in RFC3881. In total we collected 695 user access events from the distributed medical imaging system running at Mohawk College within a month. Table 6.1 indicates: attribute names; mapped attributes onto the XML schema of RFC3881; domains of attribute values; and selected minimum supports for association mining which will be further discussed in the following subsection.

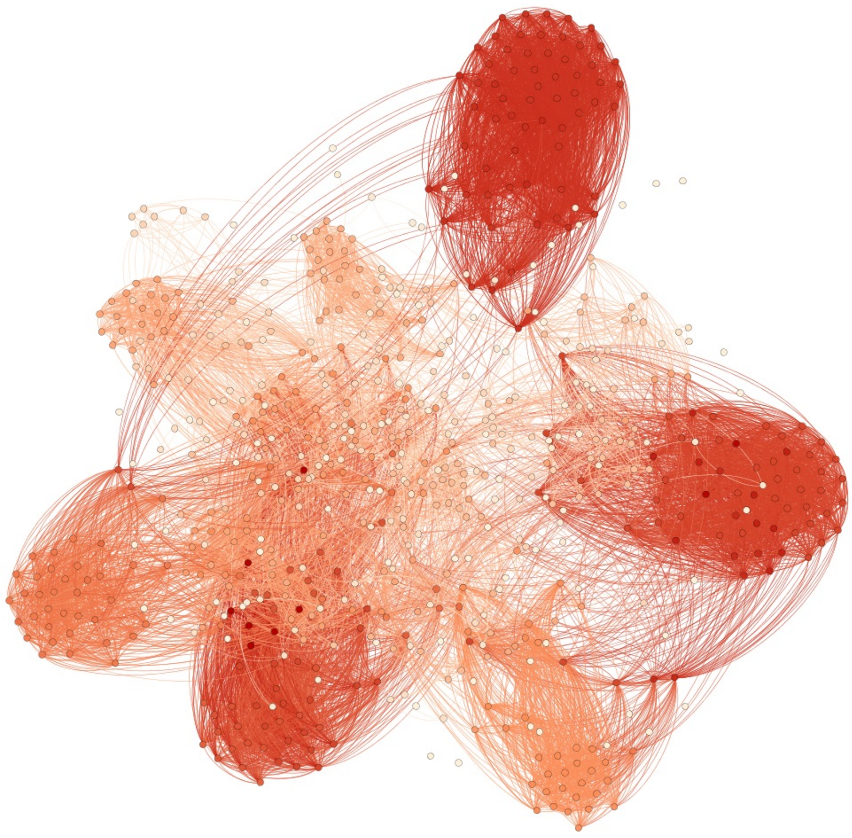


Figure 6.2: Visualization of association-based similarity between events

### 6.2.1 Maximal Association

Apiori algorithm [19] considers only one minimum support value (*minsup*) to find frequent itemset, which implicitly assumes that all items in the dataset are of the same nature and/or have similar frequencies. However, this is not the case in some real-life applications: some attribute values appear very frequently in the events, while others rarely appear but carry more significant meanings. For example, several of the collected events are related to the role of physician; but only a few of them are related to a specific patient, which are important information. If *minsup* is set too high, the patterns that involve rare attribute values will be deleted. On the other hand, if *minsup* is set too low, it may cause explosion of discovered patterns.

The method of mining association pattern with multiple *minsup* is introduced by Liu [70]. We define *minsup* for each category of attributes. The value of *minsup* of an attribute category is selected as the average probability of an attribute value inside the attribute domain. For example, the attribute domain of "Role" is 4, then the *minsup* of attribute "Role" is 0.25 since the average probability of each attribute value of "Role" is 0.25. The selected *minsup* of each attribute is presented in Table 6.1. We can see the *minsup* of attribute "Action" is high (0.3) since the audit logs just record three actions: read, update and delete. Meanwhile, only a small group of events are related to accessing the same health record. We can see the *minsup* of attribute "Resource" is very low (0.002) because there are 492 patient resources which means the probability of an event related to an patient resource is 1/492≈0.002. The method of association mining with multiple *minsup* assumes the minimum support of an itemset is the lowest *minsup* value among the items in the itemset. So the itemset "A-1, R-2" (i.e., "Action" with minsup 0.3 and "Resource" with minsup 0.002) is frequent if its support is equal or greater than *min{0.3, 0.002} = 0.002*. After applying the modified Apriori association mining algorithm with *minsup* for each attribute category, we obtained 1276 maximal association groups.

Compared with both the synthetic and real-world datasets to be analyzed in Section 6.3 and Section 6.4, the events used in this case study is relatively small. We defined an association-based similarity between two events which is explained in Section 4.2, based on the structural property of the extracted maximal association groups. The similarity between two events $e_i$ and $e_j$, denoted as $sim(e_i,\ e_j)$ is defined as the maximum of the association values between $e_i$ and $e_j$, considering that $e_i$ and $e_j$ may belong to more than one associated group $g_x$ with a different association value in each group $g_x$. Figure 6.2 illustrates the undirected graph representation of the complete event set in our case study. We used Gephi [71] which is an open source network analysis and visualization software package to illustrate the mass relationships among the events according to our defined similarity metric. Each node represents an event and each edge represents the similarity strength between two events. If a node is connected to more highly similar nodes, its color becomes heavier. Gephi also provides some layout algorithms that push out unconnected nodes or weak connected nodes ($sim(e_i,\ e_j)$ is low) far away from the rest of the graph. After applying algorithms, we obtained the view of Figure 6.2 which shows several group of events are highly associated. However, this automatic tool does not provide any information about the common characteristics of the highly associated event groups.

### 6.2.2 Constrained-clusters

We created a database of seed-domains that are used as the search domain for populating the constrained-clusters. Each event (as a potential seed) has a corresponding domain (as seed-domain). We assigned events that have non-zero similarity values with the seed into the seed-domain. Each event can be assigned to more than one domain. For each seed-domain we collected statistical data that are used for ranking. The collected statistics are: i) the number of events in the domain; ii) the average similarity value between events in the domain; and iii) the major attribute values that participate in the maximal association groups.

Seed-domain provides restricted search space for constructing constrained clusters, but creating seed-domain for each event is time consuming in large distributed systems. We prune events at the step of maximal association through increasing minimum support. The events having low association with other events are less significant so that they would be pruned and excluded from initial seed event selection.

In our experiment, after ranking the seed-domains based on domain size, event-387 is suggested as the initial seed for constrained-cluster since this domain containing 427 events is the largest domain. The itemsets shared among MAGs containing event-387 are as follows: *<U-22 P-17> <A-1 D-11> <A-1 T-10 U-22 L-6>*.

With the above suggested significant event and attribute values, the user is able to produce some constraints using our proposed BPQL discussed in Section 3.4 to construct event groups for subsequent behavior pattern mining. In order to evaluate the effectiveness of the knowledge on behavior pattern discovery result, four groups of events from loosely constrained to tightly constrained are selected as follows:

- **Cluster#1 (Without knowledge).** Collect all events to this cluster.
- **Cluster#2 (With knowledge of maximal association).** Select event-387 as initial seed event, and collect events that have non-zero similarity with the seed event. Totally 427 events are collected.
- **Cluster#3 (With knowledge of both maximal association and constraints).** Select event-387 as initial seed event, and add one intra-cluster constraint "user = U-22 && location = L-6" to this cluster. Events issued by user U-22 from location L-6, and having non-zero similarity with seed event are assigned to this cluster. Totally 156 events are collected.
- **Cluster#4 (With knowledge of both maximal association and constraints).** Select event-387 as initial seed event, and add one intra-cluster constraint "time = T-10" to this cluster. Events happened during rush hour 10:00am, and having non-zero similarity with seed event are assigned to this cluster. Totally 47 events are collected.

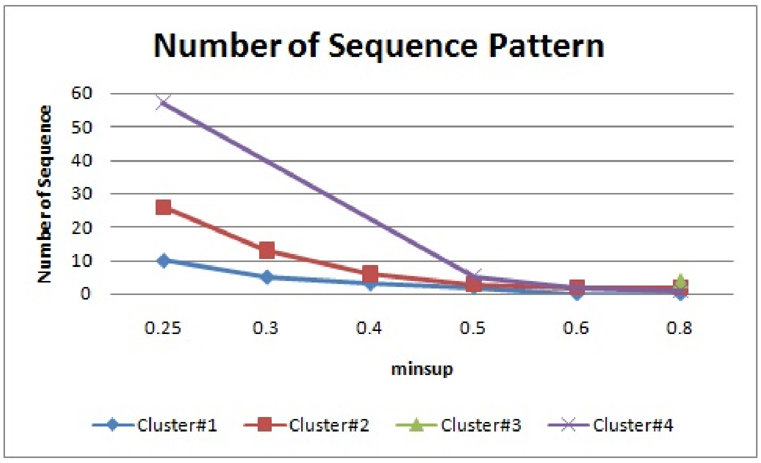


Figure 6.3: Number of discovered sequence patterns in each cluster with different minsup

### 6.2.3 Behavior Pattern Mining

Sequential pattern mining algorithm Apriori [20] is employed to discover user's daily behavior in each cluster. The input of Apriori algorithm is a sequence database and a user-specified threshold *minsup*, and the output is a list of frequent sequential patterns that occurs in a sequence database.

In our experimentation, we first convert event database to sequence database where each sequence is a set of ordered events performed by the same user within one day. Therefore, the discovered frequent sequence patterns can be viewed as the user's daily behavior. To evaluate the advantage of using maximal association groups and constraints in the process of user behavior discovery, we considered different event clusters from cluster#1 to cluster#4, and relative *minsup* values from 0.25 to 0.8. Figure 6.3 shows the number of discovered frequent sequence patterns from the four clusters. The number of sequence patterns for cluster#3 was not shown due to low values of minimum support, as it generates too many candidates and run out of memory (same reason for Figure 6.4 and 6.5). We found more sequence patterns in cluster#4 than the other clusters. This result demonstrates that the recommendation of an initial seed event and significant attribute values allow us to discover more sequence patterns.

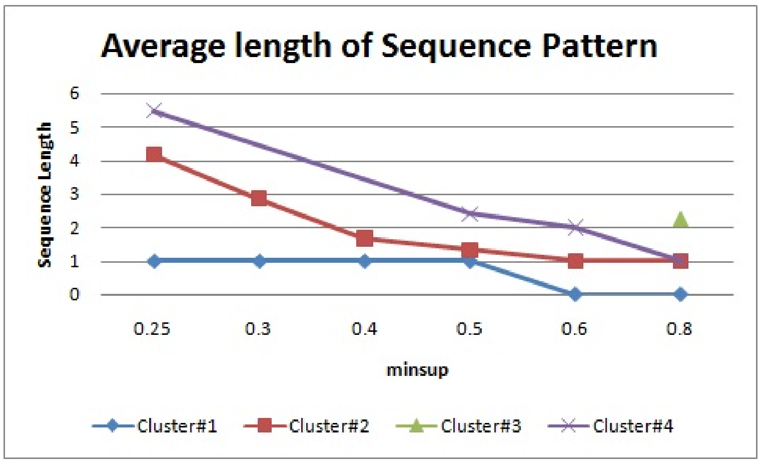


Figure 6.4: Average length of discovered sequence patterns in each cluster with different minsup

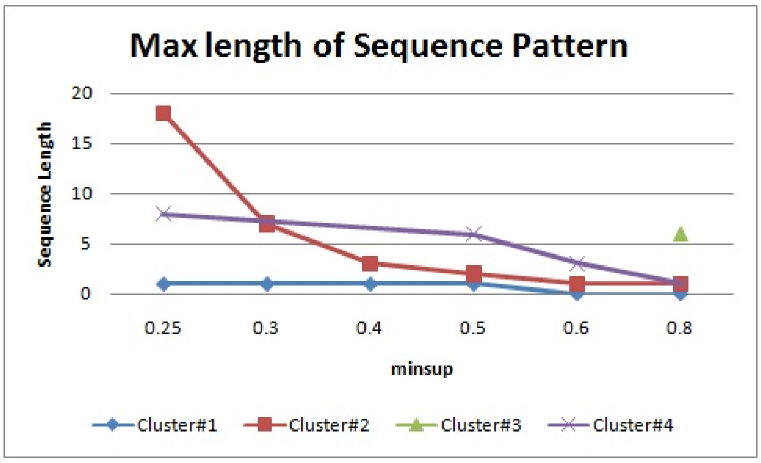


Figure 6.5: Maximum length of discovered sequence patterns in each cluster with different minsup

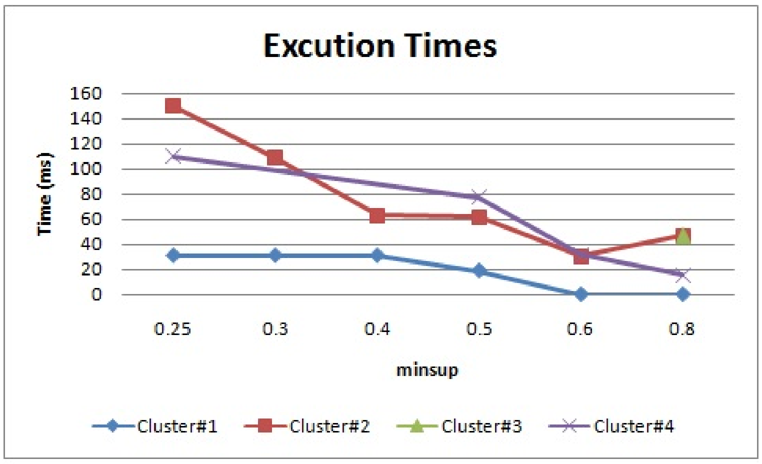


Figure 6.6: Execution time of mining each cluster with different minsup

Figure 6.4 presents the average length of discovered frequent sequence patterns for the four clusters. The length of a sequence is the number of itemsets in the sequence. A sequence of length k is called a *k-sequence*. We aim at identifying more sequence patterns and with longer lengths. Equivalently, we want to find user's behavior pattern involved more actions. We can see from Figure 6.4 that cluster#4 contains sequence patterns with higher average length for low values of minimum support, and cluster#3 contains sequence patterns with higher average length for high value of minimum support. It proves that knowledge-driven recommendations helps to discover longer sequence patterns.

Figure 6.5 demonstrates the maximum length of discovered frequent sequence patterns for the four clusters. As shown, cluster#2 contains an 18-sequence for low minimum support (0.25); cluster#4 has longer maximum length sequence pattern for medium minimum support; and cluster#3 includes 8-sequence for high minimum support (0.8) which is much longer than other clusters.

Through comparing the discovered sequence pattern number and sequence pattern length for four clusters, the experiment result proves that our proposed knowledge-driven approach, maximal association and user produced cluster constraints based on recommendations, improves the discovery of behavior patterns.

Figure 6.6 shows the execution time for mining sequence patterns from each cluster. As the minimum support decreases, the execution time of mining all event clusters increase because of the growing in the total number of candidate sequences. Very few sequence patterns are discovered from cluster#1 so that its execution time is shortest. The execution time of mining cluster#4 increases slower than mining cluster#2 because of the cluster size.

The efficiency of behavior pattern mining depends on the scale of event dataset and the minimum support parameter. The Apriori algorithm used for behavior pattern mining in our experiment is a basic algorithm but the performance is acceptable with our limited dataset size. For large distribute systems, we have evaluated some more efficient algorithms to improve the performance. After collecting more events from target system (e.g., events during a couple of months), we would apply more efficient pattern mining algorithms such as FPGrowth [72] and PrefixSpan[73]. FPGrowth is a very fast and memory efficient frequent itemset mining algorithm which uses a special internal structure called FP-Tree. PrefixSpan proposed a projection-based sequential pattern-growth for efficient mining of sequential patterns.

### 6.2.4 Behavior Pattern Analysis

In a post-analysis, based on the salient attribute values of constrained clusters the analyst investigates the characteristics of the discovered sequence patterns in each cluster. For example, what is common among the users who accessed the system around the rush hour? What is the frequent behavior pattern of a specific user in the system? Through analyzing the common attribute values in each item of sequence patterns discovered in subsection 6.2.3, context attributes are extracted to describe the circumstances of the complete sequence. Following examples are some sample behavior patterns discovered from cluster#3 and cluster#4 which assist administrators to examine the user behaviors in accessing system:

- User "U-22" at most has 6 access requests each day at location "L-6" in 80% working days; User "U-22" accesses the record of patient "P-12" at location "L-6" twice each day in 80% working days.
  *SUPPORT=0.8*
  *Actor = <U-22>*
  *Context = <L-6>*
  *Action-sequence = <<A-1> <A-1 P-12> <A-1 P-12> <A-1> <A-1> <A-1>>*
  *Time-interval= <Daily>*

- 80% of access requests from user "U-22 at location "L-6" are at time "1:00pm".

  *SUPPORT=0.8*

  *Actor = <U-22>*

  *Context = <L-6 T-13>*

  *Time-interval= <Daily>*

- 50% of access requests during rush hour "10:00am" are from user "U-22".

  *SUPPORT=0.5*

  *Actor = <U-22>*

  *Context = <T-10>*

  *Time-interval= <Daily>*

- 50% of users have access requests at most 6 times during rush hour "10:00am".

  *SUPPORT=0.5*

  *Context = <T-10>*

  *Action-sequence = <<A-1> <A-1> <A-1> <A-1> <A-1> <A-1>>*

  *Time-interval = <Daily>*

- 25% of access requests during rush hour "10:00am" are reading the records of patient "P-2" from location "L-1".

  *SUPPORT=0.25*

  *Context = <T-10 L-1>*

  *Action-sequence = < <O-2 P-2> >*

  *Time-interval = <Daily>*

## 6.3 Public Cloud Computing Services

Public cloud services provide auditing to keep track of the access of authorized users. For example, the AWS (Amazon Web Services) API call history produced by CloudTrail enables third-party tool understanding what's happening in an AWS account. In the section, we first apply sequential pattern mining on collected dataset directly, which generates a large number of patterns. It is almost infeasible to analyse the result and acquire meaningful user behavior patterns. Then we apply our approach to guide data analyst step by step: i) statistical analysis of single attribute describes the nature of the data to be analysed; ii) association measure establishes groups of highly related events and reduces search space for constrained clustering; iii) constrained clustering injects data analyst's domain knowledge into clustering process; and iv) behavior pattern mining and analysis process produces a small set of non-redundant representative user behavior patterns.

### 6.3.1 Data Collection

"Amazon AWS CloudTrail is a web service that records AWS API calls for your account and delivers log files to you. The recorded information includes the identity of the API caller, the time of the API call, the source IP address of the API caller, the request parameters, and the response elements returned by the AWS service." [74] The AWS API call history produced by CloudTrail can be used for user behavior analysis. For example, a *CreateStack* call (creating virtual machines) may result in following API calls about Amazon EC2 service (managing virtual machines) and Amazon EBS service (accessing block level storage volumes) as required by the AWS *CloudFormation* template.

AWS Identity and Access Management (IAM) service provides shared access to an AWS account. The administrator can give access to the AWS account to specific users. IAM users have their own user name and password but share the same account. We collected user activities using AWS CloudTrail of a company account for 4 months. This software company has two types of businesses: i) offering services hosting on public cloud to serve individual users; and ii) developing and

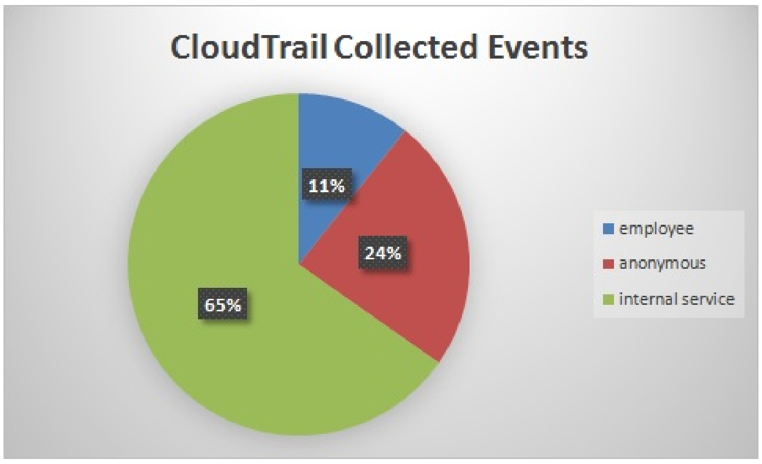


Figure 6.7: Event types recorded in AWS CloudTrail

delivering solutions for enterprise. Totally we acquired **286K** events. Figure 6.7 presents three types of events about company businesses: i) events of employees who are doing development and testing on AWS; ii) events of anonymous that record individual user accessing the company offered services; and iii) API calls by internal services. Considering the feasibility of evaluation, we applied the proposed user behavior pattern recovery process on the events of employees. Because we can show them our extracted patterns and ask them to assess the correctness.

Following is an example AWS CloudTrail event which contains fields that determine what action was requested by whom, and when and where the request was made.

```
{
  "awsRegion": "us-east-1",
  "eventID": "f4713996-669e-4f63-acc8-b81852463050",
  "eventName": "DescribeAlarms",
  "eventSource": "monitoring.amazonaws.com",
  "eventTime": "2015-09-03T15:06:20Z",
  "eventType": "AwsApiCall",
  "eventVersion": "1.03",
  "recipientAccountId": "122931797421",
  "requestID": "5556a7ec-524d-11e5-b0f8-7b8b5c2ddbea",
  "requestParameters": {
    "maxRecords": 100
  },
  "responseElements": null,
  "sourceIPAddress": "162.249.90.40",
  "userAgent": "signin.amazonaws.com",
  "userIdentity": {
    "accessKeyId": "XXXXXXXXXXXXXXXXXXXXXX",
    "accountId": "XXXXXXXXXXX",
    "arn": "XXXXXXXXXXXXXXXXXXXXX:user/john",
    "invokedBy": "signin.amazonaws.com",
    "principalId": "AIDAI44264P2REP7NJCRO",
    "sessionContext": {
      "attributes": {
        "creationDate": "2015-09-03T13:32:51Z",
        "mfaAuthenticated": "false"
      }
    },
    "type": "IAMUser",
    "userName": "john"
  }
}
```

- *awsRegion* records where the request was made to, and *sourceIPAddress* records where the request was made from. AWS currently supports 10 regions where *us-east-1* is US EAST(N. Virginia).
- *eventSource* is the service that the request was made to. For example, *monitoring.amazonaws.com* is a cloud monitoring service for AWS cloud resources and the applications running on AWS.
- *eventName* records the requested action, which is one of the actions listed in the API Reference for the services. For example, AWS allows users to configure alarms based on metrics data for resource usage monitoring purpose. *DescribeAlarms* is an action of retrieving alarm with the specified alarm name.
- *userIdentity* provides detailed information about the user who made the request.

We developed an AWS CloudTrail preprocessor to convert the collected logs into attributed events. Only employee's events are selected into the event repository to be analyzed. Finally we got **30K** events tracing the activities of 5 employees in 4 months. Table 6.2 indicates the extracted attribute names, attribute domain, and example attribute values. These five attributes are selected to learn user behavior pattern in our experiment. Attribute selection is performed by domain expert. Feature selection algorithms may be applied if the dataset has more complicated features. More attributes can be added into attributed events, such as *awsRegion* and *sourceIPAddress* if the data analyst concerns the service location and user location.

Table 6.2: Attributes converted from AWS CloudTrail Events

| **Attribute Name** | **Attribute Domain** | **Attribute Values** |
|---|---|---|
| User | 5 | William (manager)<br>John, Meng (developer)<br>David (part-time developer)<br>Ka (sales) |
| Date | 75 | from 2015-09-03 to 2016-01-08 |
| Time | 24 | 24 hours a day |
| Service | 15 | elasticmapreduce.amazonaws.com (elastic MapReduce)<br>ec2.amazonaws.com (virtual machine (VM) management)<br>s3.amazonaws.com (storage service)<br>autoscaling.amazonaws.com (auto-scale cluster capacity)<br>cloudformation.amazonaws.com (resource template)<br>... |
| Action | 147 | DescribeInstanceAttribute (retrieve VM attributes)<br>DescribeInstances (retrieve one or more VMs)<br>TerminateInstances (delete VMs)<br>GetBucketLocation (retrieve storage location)<br>CreateSecurityGroup (create a security group)<br>... |

## 6.3.2 Using Sequential Pattern Mining Directly

Sequential pattern mining is one of the most important data mining method for discovering behavior patterns. To compare the result between our approach and normal data mining method, we first applied frequent sequential pattern mining algorithm *Apriori* [20] to learn user behavior patterns directly. Each event in the event repository consists of the following fields: event-id, event-time, and the attributes to be analyzed. A user-sequence is an ordered list of events performed by the same user. Aiming at discovering user daily behavior, user-daily-sequence is defined as an ordered list of events performed by the same user within a calendar day. We convert the event repository into a sequence repository, where a user-daily-sequence consists of the following fields: sequence-id, a set of event-id, and a set of ordered events to be analyzed.

We found user may repeat some actions many times in a sequence. For example, in the scenario of searching a document on Amazon S3, a series of actions *ListBuckets* are performed until the user find the location where the target document is stored. Thereby we may see many redundant sequence patterns in the mining result such as <*ListBuckets*>, <*ListBuckets*, *ListBuckets*>, <*ListBuckets*, *ListBuckets*, *ListBuckets*>, <*ListBuckets*, *ListBuckets*, *ListBuckets*, *ListBuckets*>, etc. To solve this problem, we merge continuous actions and represent them in the format of *action+* in the prepossessing step. So that continuous *ListBuckets* is represented as *ListBuckets+*. After the sequence preprocessing, we will only see *ListBuckets* and *ListBuckets+* in the sequential pattern mining result no matter how many times they appear.

By given a *minsup* from 0.2 to 0.5, Table 6.3 presents the sequential pattern mining result on user-daily-sequence repository. We can see only 1 frequent subsequence about user sign-in is discovered with *minsup* as 0.5. We found more patterns as decreasing *minsup* threshold. However, if we set *minsup* low to 0.2, the sequential pattern mining algorithm cannot finish in 20 minutes, and the program crashes due to out of memory in the end. The mining result by given *minsup* between 0.3 and 0.4 are analysable in terms of the extracted pattern amount. But understanding and summarizing the extracted sequence patterns manually is al-

Table 6.3: Patterns discovered from AWS CloudTrail Events using sequential pattern mining directly

| **Minsup** | **Execute Time (second)** | **Pattern Count** | **Example Patterns** |
|---|---|---|---|
| 0.5 | 0.213 | 1 | <ConsoleLogin+,signin.amazonaws.com> |
| 0.4 | 0.237 | 7 | <s3.amazonaws.com+> |
| | | | <signin.amazonaws.com, ListBuckets,s3.amazonaws.com> |
| | | | <elasticloadbalancing.amazonaws.com, {DescribeAlarms,monitoring.amazonaws.com}> |
| | | | <{john,signin.amazonaws.com},john+> |
| | | | <ConsoleLogin+,signin.amazonaws.com, elasticloadbalancing.amazonaws.com+> |
| | | | <signin.amazonaws.com, elasticloadbalancing.amazonaws.com, monitoring.amazonaws.com, DescribeLoadBalancers, elasticloadbalancing.amazonaws.com> |
| | | | <ec2.amazonaws.com+> |
| 0.35 | 0.275 | 17 | ... |
| 0.3 | 138.323 | 437 | ... |
| 0.2 | >1000 | N/A | ... |

most impossible, and even worse if the mining result is huge. It is tough to answer the following questions without proper method and automatic tools. What are the most close sequence patterns? How many categories are there in the sequence patterns? What is the common features among a group of close sequence patterns? Our approach will guide and assist the data analysts in answering these questions.

### 6.3.3 Statistical Analysis

A bar chart is a graphical representation of the distribution over a categorical variable, which describes centering, dispersion (spread) and shape (relative frequency) of the data. Figure 6.8 to 6.12 show the bar charts of each attribute: user, hour, date, service and action. X-axis indicates the attribute values (not all attribute values are displayed due to space limit) and Y-axis shows how many times such

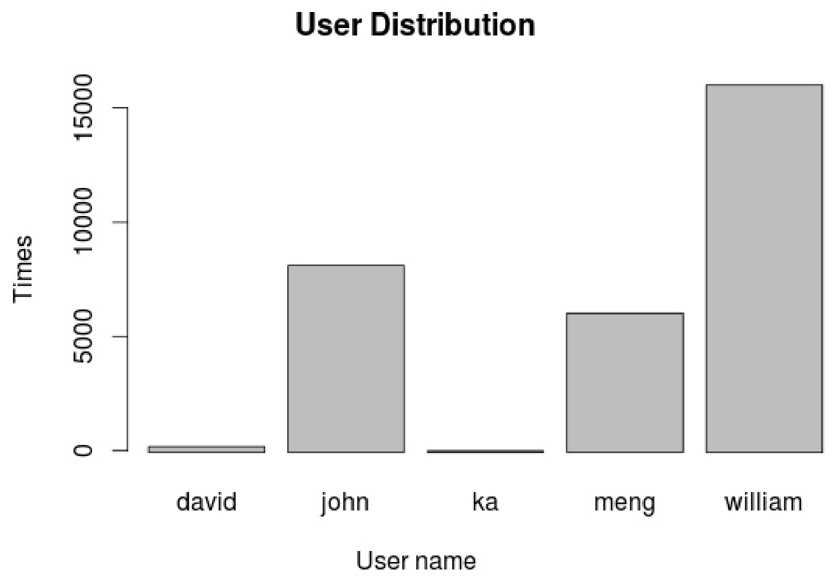


Figure 6.8: User access request distribution

attribute value appears in event dataset. Figure 6.8 reveals features of user access request distribution, such as *William* is the most active user; *John* and *Meng* take the second and third place respectively. Figure 6.9 reveals the rush hour is 16:00, and people prefer to work from afternoon until late-night (from 14:00 to 2:00). Figure 6.10 shows two obvious spikes at the first half of October, and access request amount at other days are relatively stable. About AWS services as shown in Figure 6.11, *ec2.amazonaws.com*, *elasticmapreduce.amazonaws.com*, *cloudformation.amazonaws.com*, and *s3.amazonaws.com* are the top-4 most requested services. *DescribeInstances*, *DescribeInstanceAttribute*, *DescribeCluster*, and *TerminateInstances* are the top-4 most performed actions as shown in Figure 6.12.

Consider the most frequent attributes in global, Figure 6.13 presents an overview of the most frequent attribute values where at least 5% of events in the event repository contain such attribute values. Such figures give the data analyst a first impression about the most significant individual attribute values.

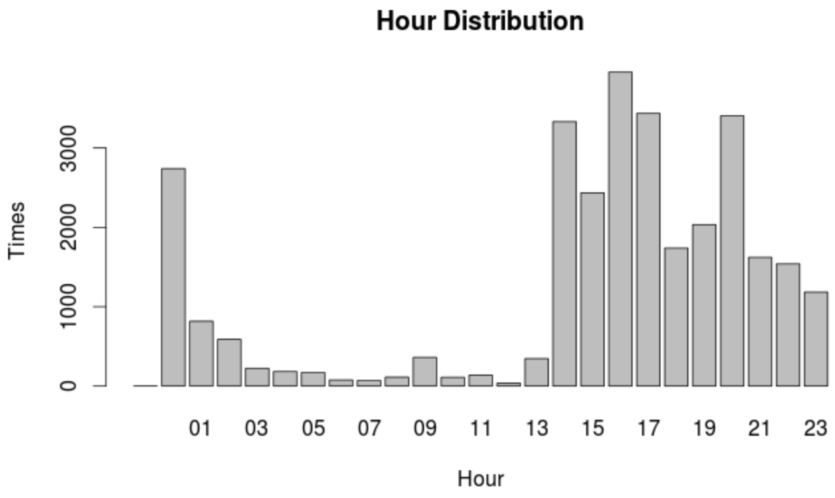


Figure 6.9: Working time distribution

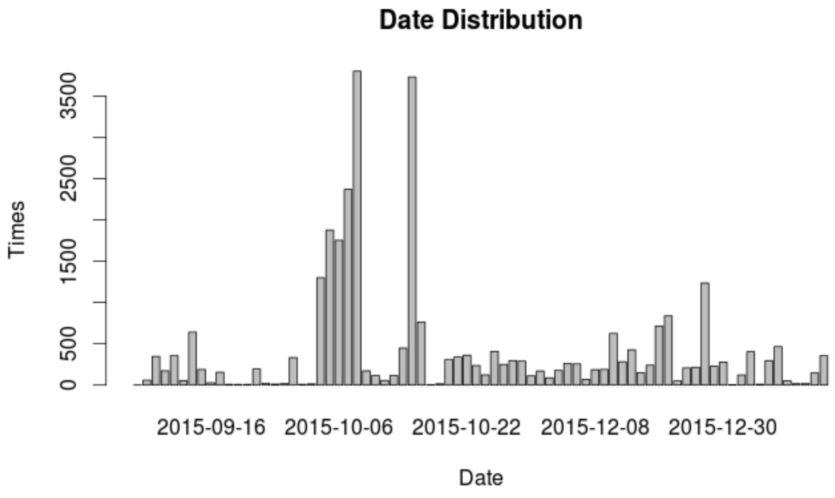


Figure 6.10: Working date distribution

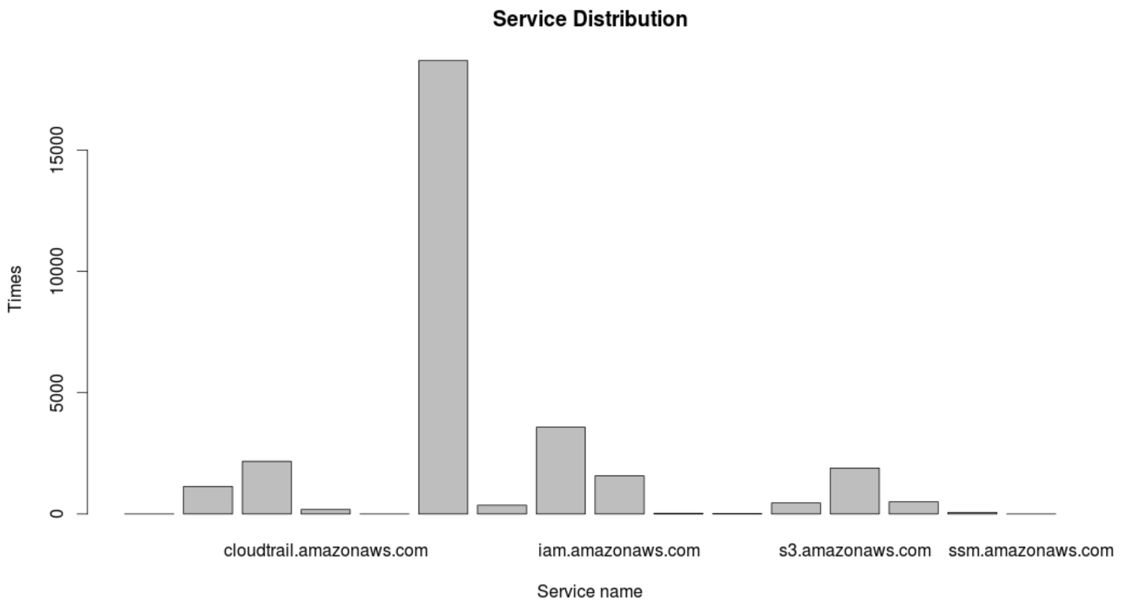


Figure 6.11: Accessed service distribution

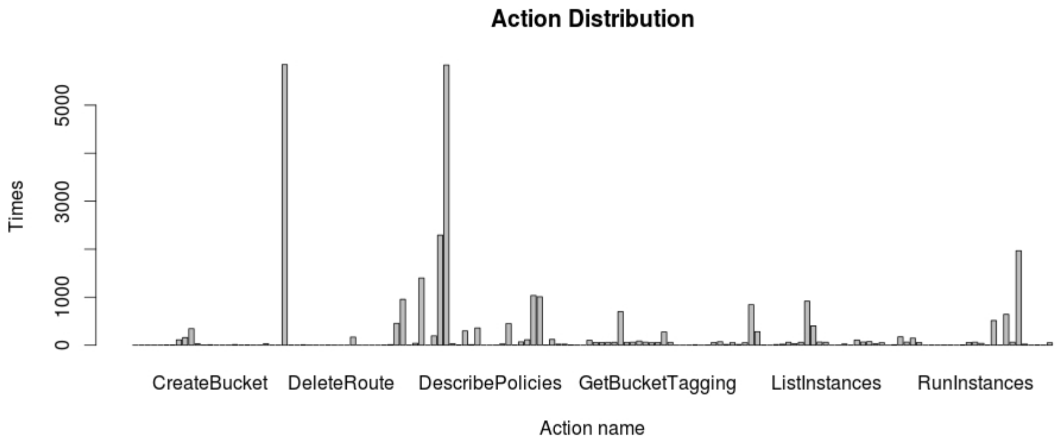


Figure 6.12: Action distribution

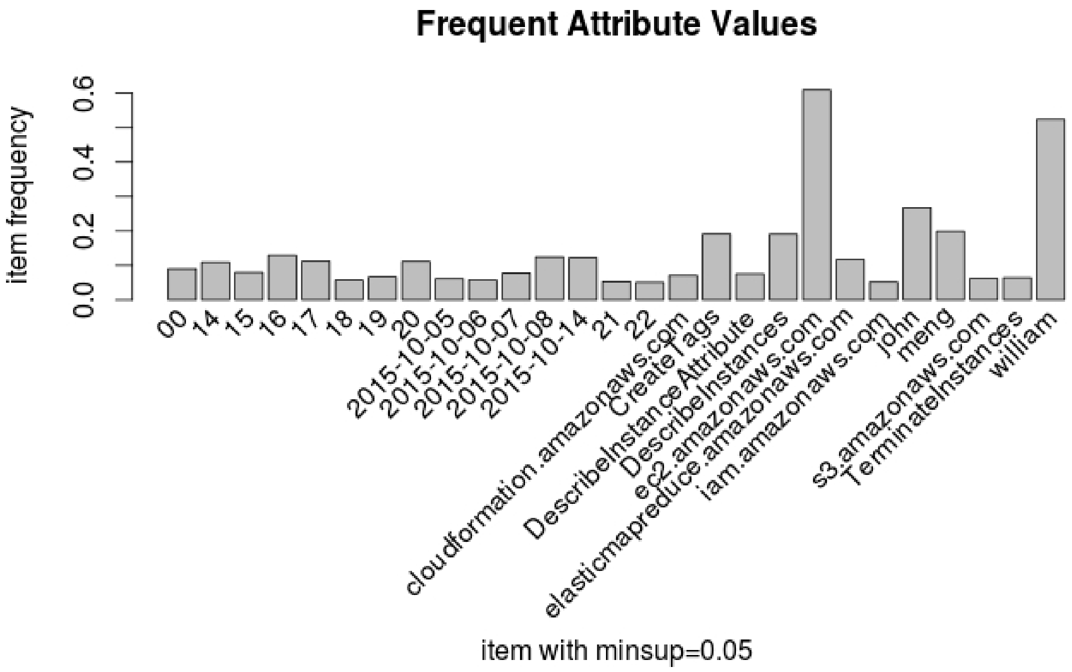


Figure 6.13: Most frequent attribute values

## 6.3.4 Maximal Association

In subsection 6.3.3, the statistical analysis of individual attribute describes the nature of the data to be analyzed. For example, the bar charts provide impressive visual display of large amount of data that helps to learn the underlying distribution, skewness, spikes, outliers, etc. But it only shows individual attribute values. Aiming to mining user common behaviors, we are also interested in discovering the relationship between entities found together. In our proposed approach, association is measured in group to establish a group of highly related events.

In Section 6.2 we applied association mining with multiple *minsup* to discover frequent patterns without losing significant rare patterns. Weight is another useful factor in representing importance of attributes in real world. Considering weights in data mining process provides a way to obtain more practical, meaningful patterns. In this case study, we assume each attribute has a weight that represents its significance when computing maximal association. For example, a MAG that groups events related to a specific service is more valuable than a MAG that groups events related to a specific date in most cases.

Maximal frequent itemset mining is an algorithm for discovering frequent maximal itemsets in a transaction database. A frequent maximal itemset is a frequent itemset that is not included in a proper superset that is also a frequent itemset. The set of frequent maximal itemsets is thus a subset of the set of frequent itemset. Maximal frequent pattern mining is a main approach to reduce search space and to remove redundant patterns, and the mining result is much less than the result of normal frequent pattern mining. We applied frequent maximal pattern mining on CloudTrail event repository, and the result is only 20% of frequent pattern mining result with lossless representation.

Table 6.4: Extracted MAGs from CloudTrail event repository with *minsup*=5%

| **MAG Id** | **Itemset** | **Itemset Length** | **MAG Size** |
|---|---|---|---|
| 1 | <john,s3.amazonaws.com> | 2 | 1653 |
| 2 | <cloudformation.amazonaws.com,meng> | 2 | 2014 |
| 3 | <elasticmapreduce.amazonaws.com,john> | 2 | 3565 |
| 4 | <2015-10-06,ec2.amazonaws.com,william> | 3 | 1752 |
| 5 | <2015-10-05,ec2.amazonaws.com,william> | 3 | 1757 |
| 6 | <ec2.amazonaws.com,TerminateInstances,william> | 3 | 1966 |
| 7 | <DescribeInstanceAttribute,ec2.amazonaws.com,william> | 3 | 2276 |
| 8 | <2015-10-07,ec2.amazonaws.com,william> | 3 | 2369 |
| 9 | <14,ec2.amazonaws.com,william> | 3 | 2732 |
| 10 | <20,ec2.amazonaws.com,william> | 3 | 1880 |
| 11 | <17,ec2.amazonaws.com,william> | 3 | 2561 |
| 12 | <16,ec2.amazonaws.com,william> | 3 | 2631 |
| 13 | <DescribeInstances,ec2.amazonaws.com,meng> | 3 | 2208 |
| 14 | <DescribeInstances,ec2.amazonaws.com,william> | 3 | 3577 |
| 15 | <00,2015-10-08,ec2.amazonaws.com,william> | 4 | 2195 |
| 16 | <2015-10-14,CreateTags,ec2.amazonaws.com,william> | 4 | 1555 |

*Maximal association* can be extracted by frequent maximal itemset mining and is considered as an interesting property for visualizing the structure of relations among groups of events. Table 6.4 presents the result of 16 MAGs after applying frequent maximal pattern mining on CloudTrail event repository with a *minsup*

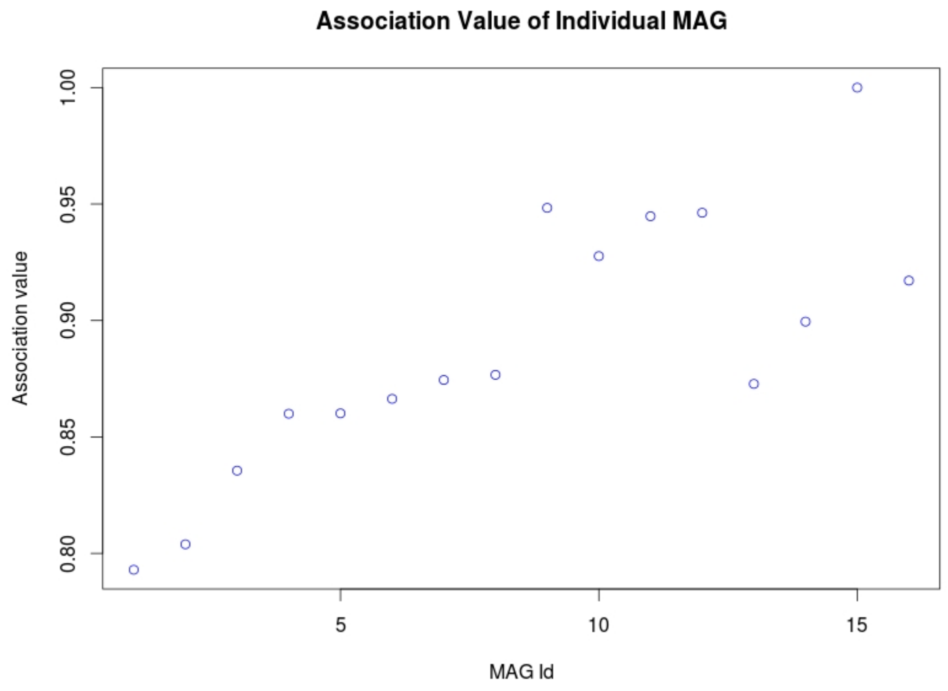


Figure 6.14: Association measure of individual MAG

of 5%. The second column of table 6.4 shows the maximal frequent itemset that is shared by all events inside the MAG; the third column indicates the length of maximal frequent itemset; and the fourth column presents how many events are collected into the MAG. We may obtain more smaller MAGs by decreasing *minsup*.

The events in a group are more associated if a large number of events in the group that share more significant attribute values. The association-based similarity measure between events in a MAG is defined in Section 4.2, which is decided by: i) the size of itemset; ii) the weight of each attribute in itemset; iii) the number of events collected into the MAG; and iv) the weight of the collected events compared with the shared attributes. We assign a weight to each attribute and a weight to collected events as shown in Table 6.5. These parameters could be optimized through analysing the pattern mining result. Figure 6.14 shows a visual display of association measure of individual MAGs according to the formula defined in subsection 4.2.1 using parameters defined in Table 6.5. The association value is normalized between 0 and 1. We can see *MAG-15* has the maximum association value.

Table 6.5: Parameters used for association measure

| **Name** | **Represent** | **Value** |
|---|---|---|
| Weight of User | $w_a \mid a = U$ | 0.3 |
| Weight of Date | $w_a \mid a = D$ | 0.1 |
| Weight of Time (hours) | $w_a \mid a = T$ | 0.2 |
| Weight of Service | $w_a \mid a = S$ | 0.3 |
| Weight of Action | $w_a \mid a = A$ | 0.1 |
| Weight of collected events | $w_c$ | 0.2 |

Each MAG represents highly similar events and is considered as the building block for behavior pattern mining. In some cases, some MAGs are also similar as they have some common events sharing the same set of attributes. Merging similar MAGs into a cluster to be analyzed may provide more opportunities to extract common behavior patterns. The association-based similarity measure between MAGs is defined in Section 4.2, which is decided by: i) the number of shared attributes between MAGs; ii) the weight of each shared attribute; iii) the number of shared events between MAGs; and iv) the weight of the shared events compared with the shared attributes. Figure 6.15 presents a visual display of association measure between two MAGs according to the formula defined in subsection 4.2.2. The size of circle indicates the relative association values. Some MAGs are similar to others evenly, such as *MAG-10* is highly associated with a set of MAGs but we cannot find the closest affinities. Some MAGs are lower associated with other MAGs but is more close to specific MAG, such as *MAG-1* is more associated with *MAG-3* compared with other MAGs.

### 6.3.5 Constrained Cluster

With the acquired knowledge by statistical analysis and association mining, the data analyst is able to produce some constraints using our proposed BPQL dis-

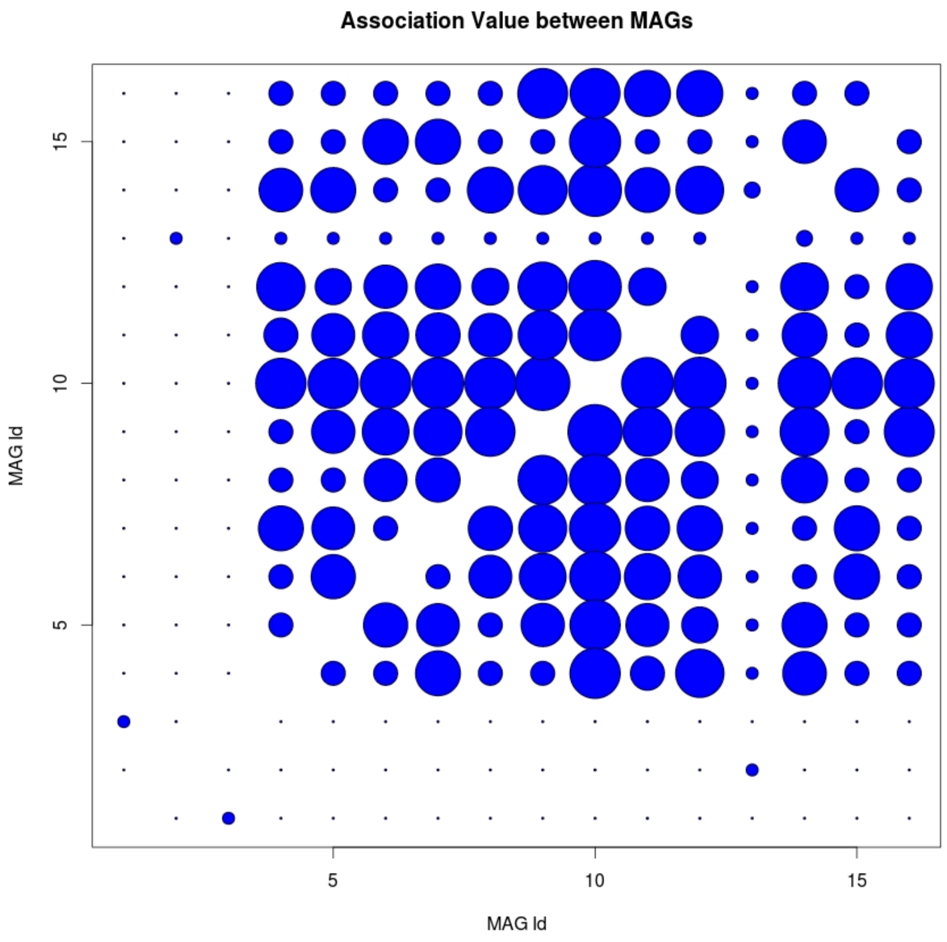


Figure 6.15: Association measure between MAGs

cussed in Section 3.4 to construct event clusters for subsequent behavior pattern mining. MAGs also provide restricted search space for constructing constrained clusters. In order to evaluate the effectiveness of the knowledge on behavior pattern mining result, three event clusters are produced as follows:

- **Cluster#1.** We know *William* is the most active user by statistical analysis. *MAG-15* gets the maximum association value as individual MAG, which contains a set of highly related events about *William* accessing service *ec2.amazonaws.com* at a specific time. Furthermore, *MAG-15* is close to *MAG-6, MAG-7, MAG-10*, and *MAG-14* according to the association measure between MAGs. With such knowledge we want to see what is the most common behavior of *William* about requesting "ec2.amazonaws.com" service for virtual machine management. *Cluster#1* collects around 5000 events about user *William* accessing to service *ec2.amazonaws.com* from *MAG-6, MAG-7, MAG-10, MAG-14*, and *MAG-15.*

- **Cluster#2.** We can see most MAGs collect events about service "ec2.amazonaws.com", and only *MAG-1* collects events about storage service *s3.amazonaws.com.* Aiming to studying the common behavior about accessing storage service, *Cluster#2* collects around 1500 events about user *John* accessing to service *s3.amazonaws.com* from *MAG-1.*

- **Cluster#3.** *MAG-1* is distinctly close to *MAG-3* compared with others even though the association value of *MAG-1* and *MAG-3* are not distinct as individual MAG. Aiming to studying the common behavior of a specific user *John*, *Cluster#3* collects around 4000 events about user *John* from *MAG-1* and*MAG-3.*

Following is the produced behavior pattern query using BPQL discussed in Section 3.4. We developed a search engine to collect events that satisfy constraints from source MAGs into constrained event clusters. *SIZE-CONSTRAINT* limits the number of events that are collected into clusters. Events are ranked based on its maximal association value. The maximal association value between two events is defined as the maximum of the association values, considering that each event may belong to more than one MAG with a different association value in each MAG. The events satisfy constraints and with higher maximal association value are assigned into the cluster until reach *SIZE-CONSTRAINT*.

```
BEGIN-BPQ
    CLUSTER := C-1;
        MAG := MAG-6, MAG-7, MAG-10, MAG-14, MAG-15;
        SIZE-CONSTRAINT := 5000;
        INTRA-CONSTRAINT
            User = 'william';
            Service ='ec2.amazonaws.com';
    CLUSTER := C-2;
        MAG := MAG-1;
        SIZE-CONSTRAINT := 1500;
        INTRA-CONSTRAINT
            User = 'john';
            Service ='s3.amazonaws.com';
    CLUSTER := C-3;
        MAG := MAG-1, MAG-3;
        SIZE-CONSTRAINT := 4000;
    INTER-CONSTRAINT
        C-2.User = C-3.User;
END-BPQ.
```

### 6.3.6 Behavior Pattern Mining and Analysis

We convert each event cluster to sequence cluster where each sequence is a set of ordered events performed by the same user within one day. Then we apply frequent maximal sequential pattern mining on each sequence cluster with different input parameter *minsup*. The discovered frequent subsequence patterns from each sequence cluster can be viewed as user's daily behavior. Following is behavior pattern mining result of each constrained cluster.

#### Cluster #1

By given a *minsup* as 0.8, the frequent maximal sequential pattern mining algorithm extracted more than 9K subsequence patterns from *Cluster #1* which is very difficult to be analysed by human (patterns are not displayed due to the large number). In this case, to reduce the extracted pattern number the data analyst may add more cluster constraints to reduce the cluster size or shorten the user sequence time interval such as user's hourly sequence or session sequence. At the same time we found user *William* has a large number of unusual access requests at specific days from *Cluster #1*. Figure 6.16 shows the access request distribution of *William*. At the first week of October and October 14th, *William* had thousands access requests each day. Such user behavior pattern is suspicious as it causes thousands dollar bill. The data analyst confirmed with user *William* himself, he was doing scalability and stress test of a distributed cluster which requires hundreds of virtual machines running for a long time. That explained why *William* has large number of access request at specific days. So these unusual behavior patterns are normal at specific software test stage.

#### Cluster #2

By given a *minsup* as 0.4, the frequent maximal sequential pattern mining al-

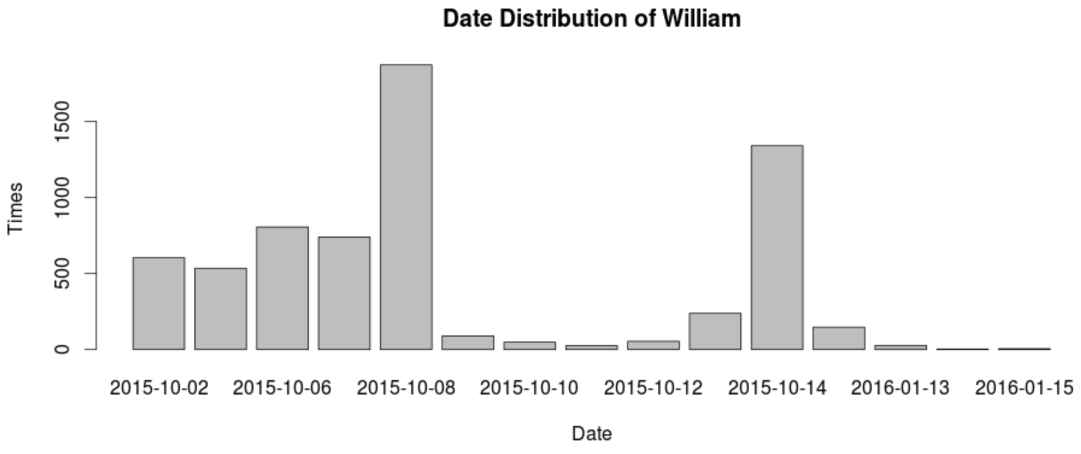


Figure 6.16: Date distribution of user william

gorithm extracted 40 action subsequence patterns from *Cluster #2*. Figure 6.17 provides a visualization of user *John*'s frequent action sequences when accessing Amazon S3 service. X-axis *t0, t1, ..., t4* means a step of the sequence, and Y-axis represents the discovered sequence number such as 1 means the first frequent sequence pattern. Each colourful rectangle represents one single action; each row is a frequent sequence pattern. For example the second sequence pattern is *<ListBuckets+, GetBucketLocation+, GetBucketVersioning, GetBucketLocation+>*, which describes a typical behavior of searching documents/resources stored at Amazon S3 service. The discovered sequence patterns are meaningful and important for analyzing the common behavior related to Amazon S3 service. In contrast, the discovered patterns in subsection 6.3.2 are dispersed over various Amazon services, such as signin service, S3 service, load balancing service, monitoring service and ec2 service, which is very hard to be analyzed and interpreted. However, the patterns discovered from *Cluster #2* are all related to a specific service and a specific user. The task of analyzing the discovered patterns from *Cluster #2* is to extract and summarize how user *John* utilizes S3 service.

The discovered 40 patterns are the most significant sequences existing in **Cluster #2**, but we cannot answer following questions from Figure 6.17. Can these

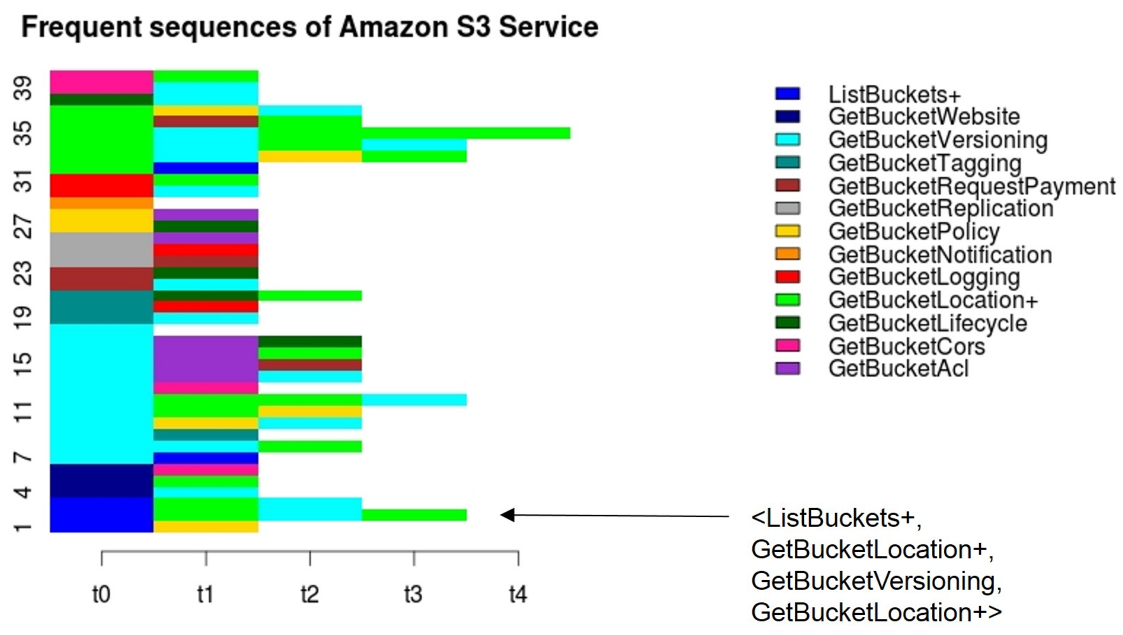


Figure 6.17: Discovered 40 frequent action sequence patterns of Amazon S3 Service. Each colourful rectangle is an action of sequence; each row is a frequent sequence pattern; t0, t1, ..., t4 means one step of the sequence.

patterns be categorized? Which patterns are more similar than others? What are the common characteristics among similar patterns? We may get answers by dividing the sequence patterns into a number of clusters and exploring the representative patterns of each cluster. Clustering is one of the most common methods of grouping similar entities. Representative pattern mining helps to remove the redundant patterns from each cluster, and finally helps to acquire a small group of most significant patterns with least representation loss.

Figure 6.18 indicates the hierachical clustering result on discovered 40 frequent sequence patterns. Agglomerative hierarchical clustering is a bottom-up approach where initially each observation is a cluster by itself; then pairs of nearest clusters are merged as one until only one cluster containing all observations remains [75]. The distance between two sequences are measured on longest common subsequence (LCS). After producing the hierarchy tree, clustering process is done by cutting the tree at a given height according to desired cluster number [76]. We can see

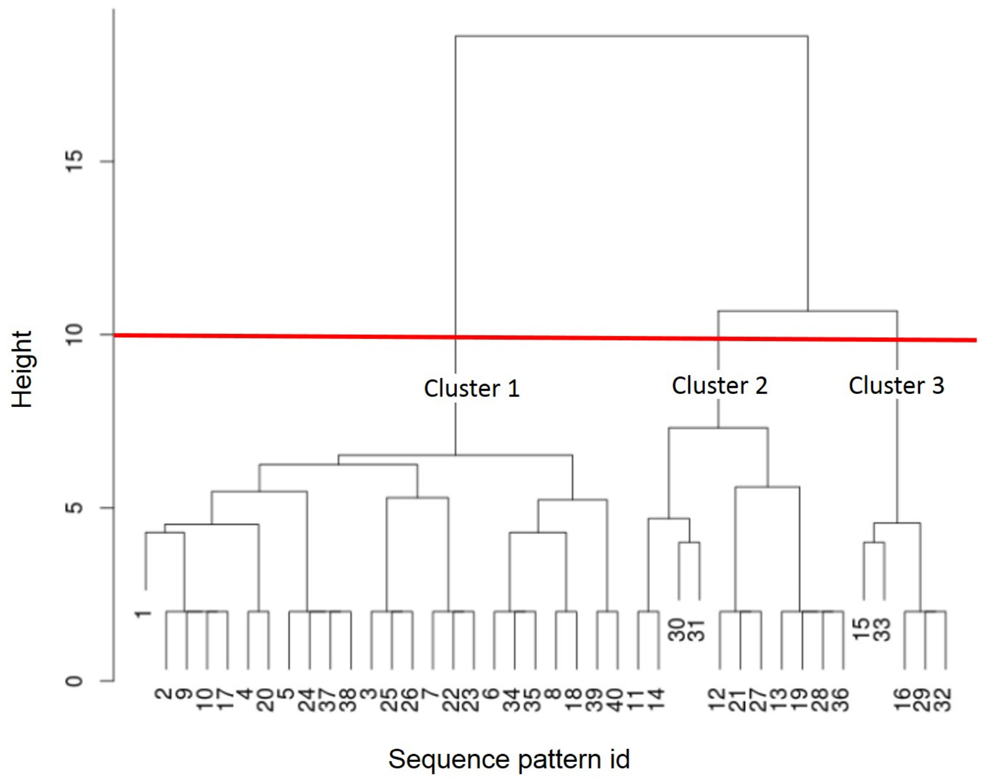


Figure 6.18: Hierarchical clustering of frequent action sequence patterns from Figure 6.17

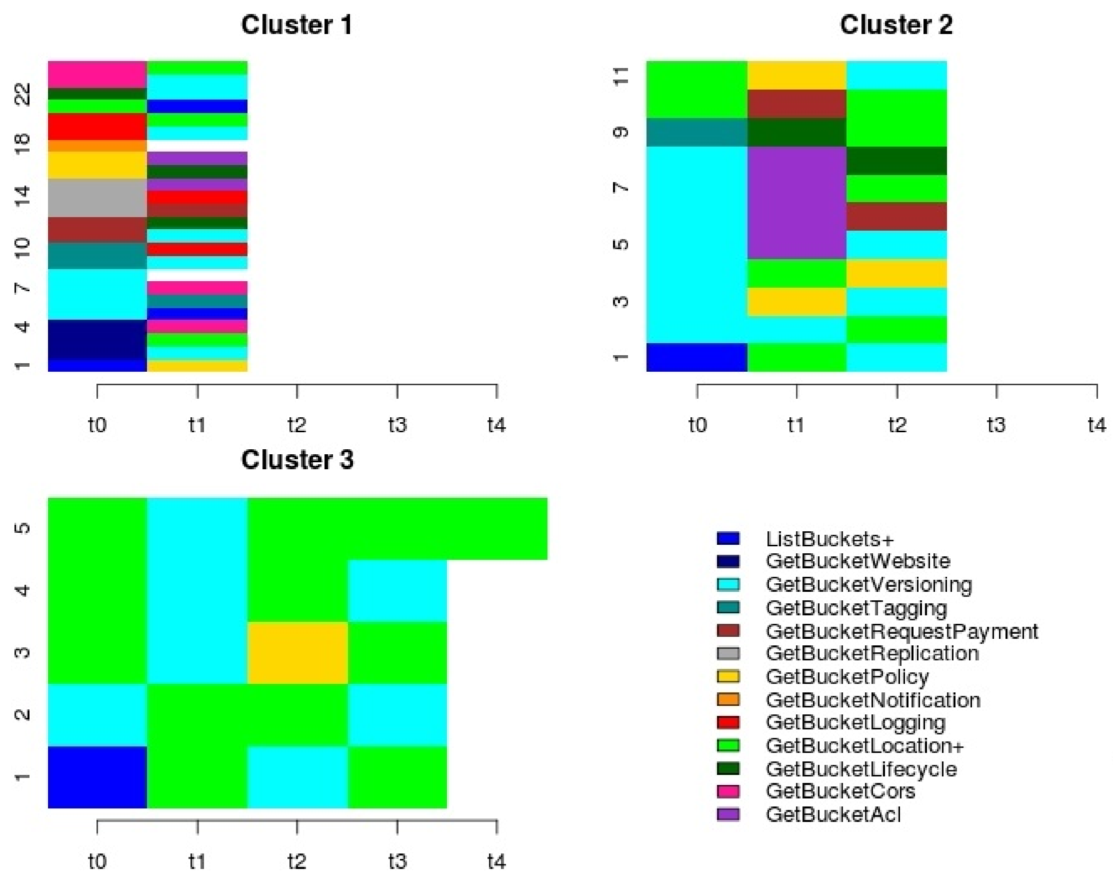


Figure 6.19: 40 action sequence patterns are divided into 3 clusters based on hierarchical clustering algorithm shown in Figure 6.18

from Figure 6.18 that the best choice for total number of clusters are 3. Figure 6.19 shows the action sequence patterns assigned to three clusters.

Extracting representative sequences is a process of mining as small as possible set of non-redundant sequences covering a desired percentage of all sequences. A sequence covers another sequence if they are similar. Representative sequence extraction algorithm is applied on each cluster of frequent sequence patterns in Figure 6.19. The representative sequences are obtained by an heuristic algorithm: i) each sequence is assigned a representative score according to neighborhood density; ii) all sequences are sorted in descending order by assigned representative score which constructs the candidate list; iii) representative sequence is selected from the head of candidate list if it is not redundant with already retained rep-

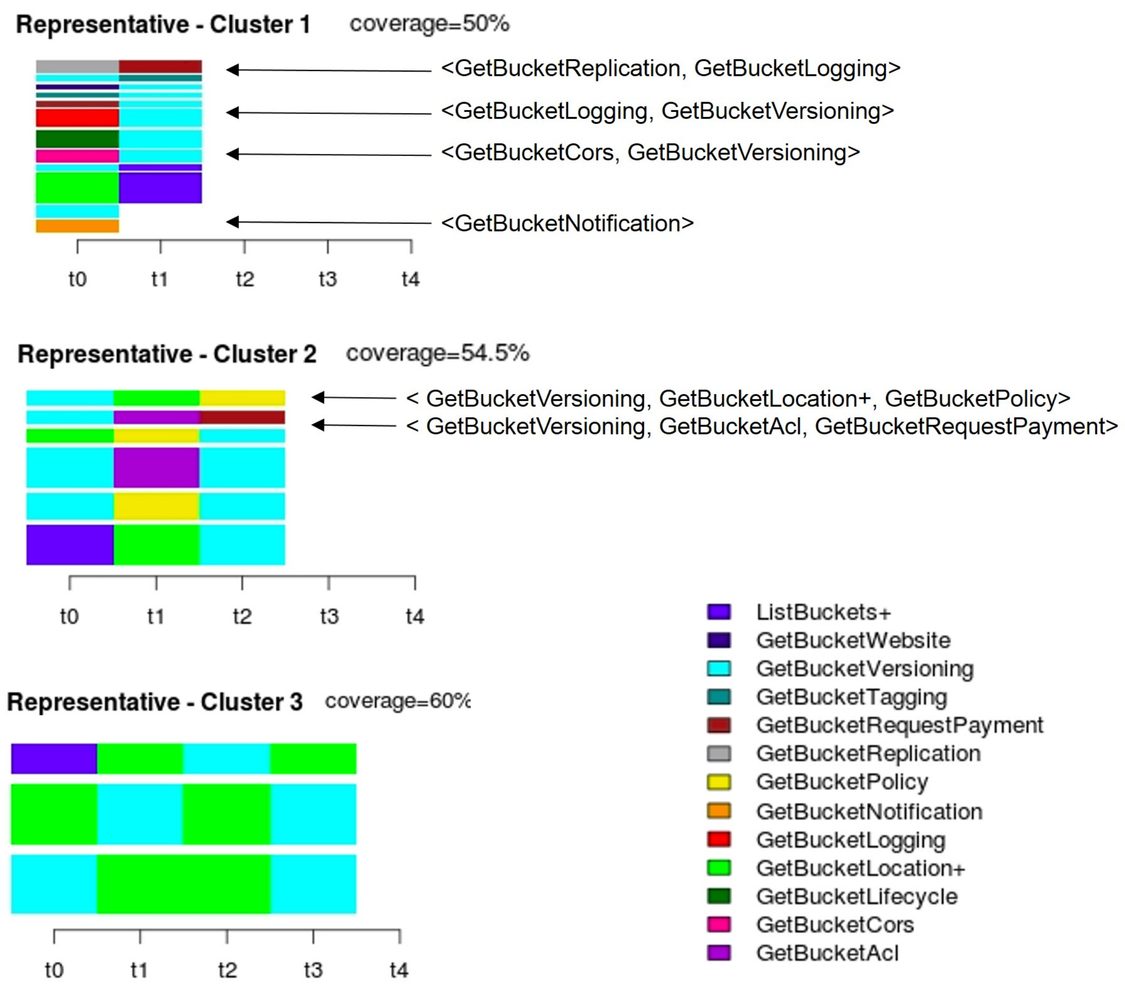


Figure 6.20: Representative sequences discovered from clusters in Figure 6.19 with minimum coverage of 50%

resentative sequences; and iv) the selection stops until the desired coverage is reached.

Figure 6.20 presents a set of non-redundant representative sequences of each cluster with at least 50% coverage. After checking the Amazon API specification, we can see these three clusters include some behavior patterns about real use cases.

- **Monitoring.** *cluster-1* shows the behavior patterns about storage monitoring: i)*GetBucketNotification* reads the bucket notification where a bucket is a cloud storage; ii) GetBucketCors reads the configuration about bucket cross-origin resource sharing; iii) *GetBucketLogging* checks the bucket logging status; and iv) *GetBucketReplication* checks bucket replication for high-availability purpose, etc.

- **Permission Management.** *cluster-2* indicates the behavior patterns about storage access permission management: i) *GetBucketPolicy* reads the policy of specified bucket; ii) *GetBucketAcl* checks the bucket access control list; and iii) *GetBucketPayment* reads the payment configuration of a bucket, etc.

- **Searching and Querying.** *cluster-3* includes behavior patterns about bucket searching and querying.

### Cluster #3

*Cluster #3* collects highly related events about user *John*. Instead of analyzing the behavior of accessing Amazon service as we did in analyzing *Cluster #2*, we want to see if there is any pattern related to *John*'s working time. By given a *minsup* as 0.15, the frequent maximal sequential pattern mining algorithm extracted 7 time sequence patterns from *Cluster #2* as shown in Figure 6.21. *02+* means user *John* has more than 1 access request on Amazon service at 2:00am. The first sequence pattern at the bottom of the figure is *<02+, 03+, 04+, 05+>*, which means user *John* has more than 1 access request on Amazon service at 2:00am, 3:00am, 4:00am and 5:00am respectively. The maximum length of the frequent time sequence patterns is 4, and the minimum length is 1. It shows *John* sometimes continuous being working on Amazon for four hours, but mostly he works for one or two hours consecutively.

Also using LCS to measure sequence similarity and hierachical clustering algorithm, these frequent time subsequences are divided into 2 clusters as shown in Figure 6.22. After confirming with user *John*, the extracted patterns indeed have reasonable meanings.

- **Preferred Working Time.** *cluster-1* shows the behavior patterns about *John*: i) he can spend more time on development in the morning from 09:00

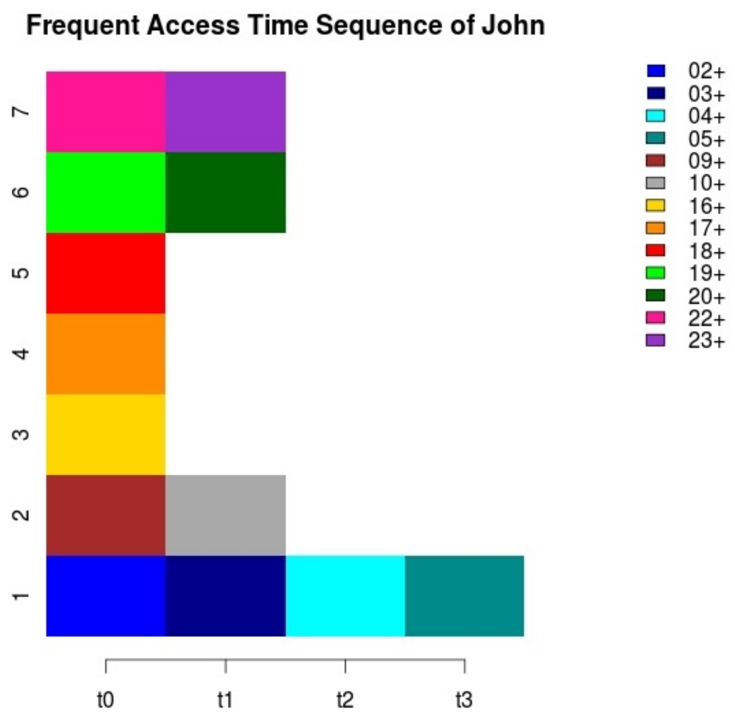


Figure 6.21: Frequent time sequence pattern of user John

to 10:00, and at night from 19:00 to 23:00; and ii) even though the time from 16:00-18:00 is the rush hour for most employees of the company as shown in Figure 6.9, *John* normally has meetings with customer in the afternoon so that focusing on development work for successive hours is difficult for him.

- **Testing Program.** *cluster-2* indicates the behavior patterns about a testing program running from 02:00 to 05:00. Amazon spot virtual machines are often available at a discount during idle time such as midnight.

## Conclusion

The conclusion of this case study is as follows: i) without priori knowledge, MAGs provide recommendation of significant attributes and highly related events in group; ii) ranking group of events by association measure helps to reduce search space with less significant information loss; iii) sequence clustering and representatives extraction summarizes the large number of frequent sequence patterns; and iv) compared with applying sequential pattern mining method directly, our pro-

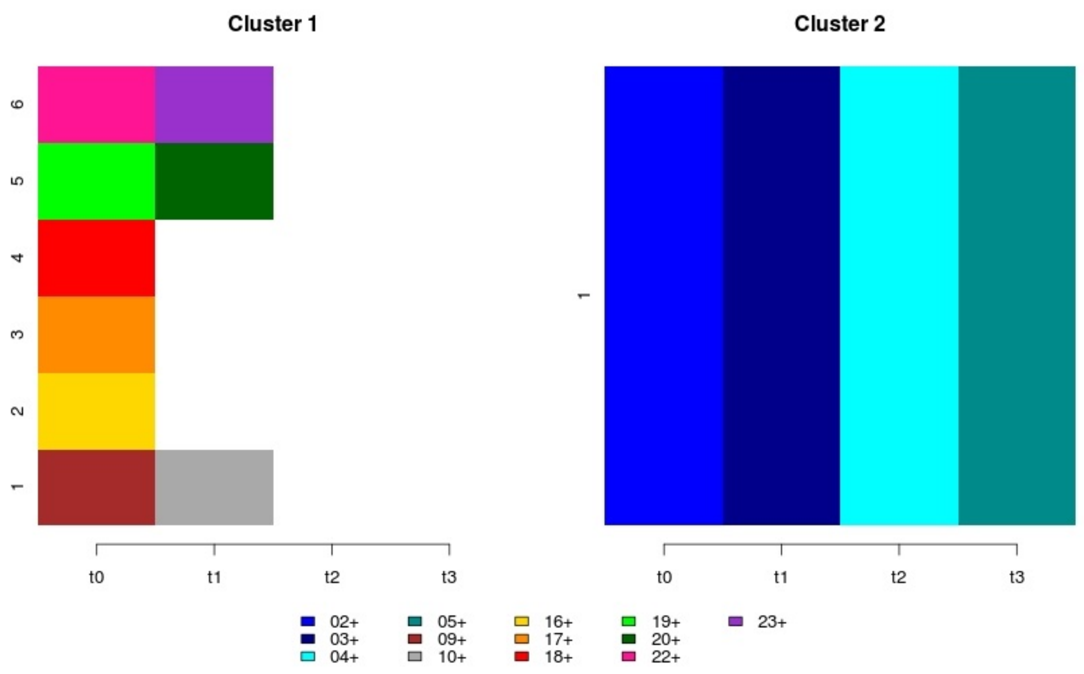


Figure 6.22: Clustering result of frequent time sequences from Figure 6.21

posed approach is more efficient and capable of guiding data analyst in discovering more meaningful user behavior patterns.

Besides, our approach detects unusual resource usage pattern in cluster-1: *William* has large number of access request at specific days, which costs thousands of dollars. This discovered pattern is quite different from his previous behavior; also such behavior is suspicious to a small software company. Even though this behavior is regarded as normal at last because it happens at specific software test stage, the discovered patterns still provide useful knowledge to system administrators.

## 6.4 Synthetic Dataset

In Section 6.2 and 6.3, we applied our approach on two real-world datasets, one from medical imaging system and the other from public cloud computing service, which prove our approach helps the data analysts in discovering more meaningful user behavior patterns. Moreover, a labelled dataset is required to evaluate our behavior pattern mining result quantitatively. Without normal labels we cannot evaluate our approach nor compare with others. However, getting labelled datasets from real-world system is difficult and time consuming. Labels for datasets are often obtained by asking domain expert to make judgements about small pieces of unlabelled data, which is too expensive and even infeasible for discovering user behavior patterns from large number of events. Thus, we created a dataset using our *EventGenerator* tool to insert meaningful user behavior patterns into the generated event dataset, along with proper labels to the inserted behavior patterns.

### 6.4.1 Dataset Design

We created an event dataset to simulate real medical imaging system, where each event records the following information: user name, role assigned to the user, user location, requested service, action, accessed patient, date, and time. Table 6.6 indicates input parameters of *EventGenerator* tool, specifying the desired attribute value distribution in generated event dataset. The attributes and their distributions are designed after analysing the characteristics of real-world event logs used in Section 6.2. The detailed attribute definition is explained in Chapter 5. *EventGenerator* randomly selects attribute values but follows the specified distribution in generation process. Appendix B presents generation result: the histograms of attribute distribution extracted from generated event dataset.

We defined 10 typical user behavior patterns in different scenarios, including: i) 3 behavior patterns about nurse wand-round between locations; ii) 3 behavior patterns about radiologist workflow with different action steps; and iii) behavior patterns about accessing request time sequence. Each behavior pattern definition specifies behavior actor, behavior context, behavior sequence, behavior time constraints, and behavior support. Appendix B presents the entire behavior pattern

Table 6.6: Attribute distribution definition of synthetic dataset

| Attribute | Representation | Domain | Type | Mu | Sigma |
|---|---|---|---|---|---|
| User | U- | 100 | Random | - | - |
| Role | R- | 13 | Normal | 5 | 6 |
| Location | L- | 15 | Normal | 8 | 6 |
| Service | S- | 14 | Normal | 11 | 6 |
| Action | A- | 16 | Normal | 10 | 10 |
| Patient | P- | 300 | Normal | 150 | 50 |
| Date | D- | 60 | Random | - | - |
| Time | T- | 24 | Normal | 11 | 6 |

definition in JSON format. By adding one more parameter, average event number per user per day as 15 to control the size of the generated dataset, *EventGenerator* tool generated 45,000 events totally with predefined user behavior patterns embedded.

### 6.4.2 Result Measure and Compare

Precision and recall are the basic measures used in pattern recognition and evaluating search algorithms [77]. In our case, precision measure reflects how useful the user behavior pattern mining result are, and recall measure reflects how complete the result are. Suppose the user defined behavior patterns are relevant patterns, precision is the ratio of the number of relevant patterns discovered to the total number of extracted patterns; recall is the ratio of the number of relevant patterns discovered to the total number of relevant patterns. F-measure [77] conveys the balance between precision and recall, which is defined as following:

$$F = 2 * \frac{precision * recall}{precision + recall}$$

Consider our generated event dataset containing 10 user behavior patterns, and 20 patterns are discovered by a behavior pattern mining approach. If 5 of the discovered patterns are relevant patterns, but 15 are meaningless patterns generated randomly. Then the precision is 5/20=25%; the recall is 5/10=50%; and the F-measure is 2*(25%*50%)/(25%+50%)=33%.

Table 6.7 shows the behavior pattern mining result evaluation of different approaches. The first approach *seq-mining* applies frequent maximal sequential pattern mining on the whole dataset with different minimum support, which is the normal method used for determining user behavior in sequence database. Due to the lack of focus and the large dataset size, thousands of sequence patterns are discovered but none is relevant. If we continue to decrease *minsup* until relevant patterns are extracted, the large number of irrelevant patterns makes the result is very difficult to be analyzed in practice. In contrast, our proposed approach applies association mining *assoc-mining* on the whole dataset to extract common behavior context first. As specified in Appendix B, 3 common behavior context are defined which are relevant patterns. With *minsup* as 0.05, our approach extracted 2 patterns and all are relevant patterns; with *minsup* as 0.015, our approach extracted 20 patterns and all relevant patterns are included in the result.

3 event clusters are produced by selecting events related to the extracted behavior pattern context (*assoc-mining* result) into clusters. The events in cluster are highly related by sharing attribute values. After converting event cluster to user's daily sequence cluster, we apply frequent maximal sequential pattern mining *seq-mining* on each cluster. Table 6.7 shows the extracted sequence patterns from each cluster with different *minsup*. As decreasing *minsup*, all relevant sequences are discovered with relatively small amount of irrelevant patterns. In the worst case of our experiment (cluster2 sequence mining), if all relevant patterns are contained in the result, $417 - 3 = 414$ irrelevant sequences patterns are extracted from cluster1. But compared with the large number of sequence patterns discovered by *seq-mining* directly (more than 10,000), our approach significantly improves the mining result both in accuracy and in completeness.

Table 6.7: Behavior pattern mining result evaluation using synthetic dataset

| **Approach** | **minsup** | **Retrieved Patterns** | **Embedded Patterns** | **Relevant Patterns** | **Precision** | **Recall** | **F-measure** |
|---|---|---|---|---|---|---|---|
| seq-mining directly | 0.6 | 335 | 10 | 0 | 0 | 0 | 0 |
| | 0.5 | 920 | 10 | 0 | 0 | 0 | 0 |
| | 0.4 | 2715 | 10 | 0 | 0 | 0 | 0 |
| | 0.3 | 9875 | 10 | 0 | 0 | 0 | 0 |
| assoc-mining | 0.05 | 2 | 3 | 2 | 100% | 66.7% | 79.9% |
| | 0.015 | 20 | 3 | 3 | 15% | 100% | 26.1% |
| cluster1 seq-mining | 0.8 | 2 | 3 | 1 | 50% | 33.3% | 40% |
| | 0.2 | 106 | 3 | 2 | 1.89% | 66.7% | 3.8% |
| | 0.1 | 417 | 3 | 3 | 0.72% | 100% | 1.4% |
| cluster2 seq-mining | 0.4 | 21 | 3 | 1 | 4.8% | 33.3% | 9.5% |
| | 0.3 | 45 | 3 | 2 | 4.4% | 66.7% | 8.7% |
| | 0.2 | 101 | 3 | 3 | 3.0% | 100% | 5.8% |
| cluster3 seq-mining | 0.6 | 6 | 4 | 2 | 50% | 50% | 50% |
| | 0.4 | 18 | 4 | 4 | 22.2% | 100% | 36.3% |

### Conclusion

The conclusion of this case study is as follows: i) our proposed approach has high recall but low precision; ii) recall is more important than precision to system administrator in security analysis as missing any user behavior pattern may cause negative impact; and iii) compared with applying sequential pattern mining directly, our proposed approach significantly reduced the number of irrelevant patterns in mining result.

# Chapter 7

# Conclusion and Future Work

## 7.1 Conclusion

This thesis contributes to the user behavior pattern analytics research area by providing a recommendation system for interactive user behavior pattern recovery. User behavior analytics helps organizations to explore user behavior patterns, and then apply algorithms and statistical analysis to detect meaningful anomalies. However, the security analysts in large distributed systems are overwhelmed by the number of system users, the complexity and changing nature of user activities. Lacking a general user behavior pattern model restricts the effective usage of data mining techniques. Limited access to real world audit logs due to privacy concerns also blocks user behavior analytics development (Chapter 1).

A behavior pattern is consistent observations of a sequence of actions that a user or a group of users conducted in a common context during a specific time interval (e.g., an hour, a day, a week). Accordingly, a new behavior model (Chapter 3) is defined as a combination of sequencing, association and timing rules, which is generic, system independent and configurable based on the target application domain. An interactive and iterative method based on several data mining techniques (Chapter 2) is proposed that can efficiently discover user's common behavior pattern from unlabeled dataset. A behavior pattern query language is also proposed that allows data analysts to describe the high level and abstract event clusters for behavior pattern mining (Chapter 3). A formal specification is introduced to formalize common behavior pattern mining system using Z notation (Chapter 4). Formalism effectively helps to abstract and revise the system design.

Without prior knowledge, the thesis applies association mining for discovering knowledge such as the most associated events and the interesting attributes. It is intended to bring together highly related events, and the shared attributes that indicate the common context of the group of events. Maximal association is considered as an interesting property for visualizing the structure of relations among groups of entities. We use the notion of maximal association to define two similarity measures (Chapter 4): similarity measure between events inside a maximal association group; and the similarity measure between two maximal association groups.

There is a general lack of access to real-world audit logs, and in particular in sensitive and mission-critical industries. Moreover, finding or constructing a useful dataset from real-world systems is difficult due to the nondeterministic nature of the real-datasets. A prototype toolkit named *"EventGenerator"* has been developed to synthesize dataset in different application domains (Chapter 5), which assists data analysts in designing and producing behavior-based dataset with embedded behavior patterns. *"EventGenerator"* is used as a benchmark for evaluating the quality of user behavior pattern discovery result (Chapter 6).

Experimentation is presented with two real-world datasets from the medical imaging and public cloud service domains respectively, as well as a synthetic event dataset with embedded user behavior patterns (Chapter 6). The experiments are divided into three case studies to demonstrate the functionality, performance and accuracy of the proposed user behavior pattern recovery technique. The case studies of analyzing real-world datasets prove that our proposed approach is generic to be applied into different application domains. Moreover, compared with applying sequential pattern mining method directly, our proposed approach is more efficient and capable of guiding data analyst in discovering meaningful high-quality patterns. The case study of analysing synthetic dataset shows that our proposed approach has high recall but relatively low precision. Recall is more important than precision to security analysts as missing any user behavior pattern may cause negative impact.

## 7.2 Limitation of the Approach

The proposed approach in this thesis has the following limitations that could form the basis for further research:

- The proposed approach is limited to discover independent and deterministic behavior patterns. This approach currently cannot identify cooperative user behavior pattern such as in a specific scenario a group of users cooperate with each other on a sequence of tasks. Also the proposed behavior model is deterministic, then this approach cannot discover non-deterministic patterns.

- This thesis work is based on an assumption that frequent behavior is common behavior. Thus this approach cannot detect rare but normal behavior patterns. For example, in medical systems the users may acquire higher priority to access resources in emergency situation. If the emergency scenario happens much less than others, this approach will fail to detect behavior patterns under such scenario.

- The abstract behavior of *EventGenerator* is based on our proposed behavior model, where a behavior pattern is defined as a combination of association, sequencing, timing rules and frequency. Thus this tool currently is limited to evaluate behavior mining approaches that are built on this behavior model. But if the behavior model is enriched in future work, such as being able to represent cooperative behavior and non-deterministic patterns, a plugin of *EventGenerator* may be developed to ensure such complicated patterns with new features being inserted into generated dataset.

## 7.3 Future Work

Holistic behavioral solutions to discover user access behavior patterns and then detect ongoing insider threats is timely and promising, especially with the rapidly developing techniques of cloud computing and big data. In cloud environment, perimeter defence is not feasible. The increased use of cloud based applications exposes data in a risky environment. Moreover, the exploding data growth is leaving the protected boundary. Possible extension to the work presented in this thesis may focus on following areas:

- This thesis presents an approach to assist system administrators in exploring user common behavior patterns by analyzing event logs. The extracted common behavior patterns might be used for anomaly detection by comparing with user's dynamic behavior. We may consider a hypothesis to define behavior anomaly: a behavior anomaly is an observation that is considerably dissimilar to or inconsistent with individual's history behavior, or dissimilar to common behavior of his peer group. If an individual performs quite differently from his previous behavior, his current behavior is suspicious. If a person is categorized by role, he is supposed to perform similarly with the people who are assigned the same role. If a person has a collection of neighborhoods who are sharing the same context, he is expected to behave similarly with these neighborhoods. If a person behaves quite different from his peer, or different from the neighborhoods under the same context, his current behavior is also suspicious.

- Automation of above anomaly detection process is required when deploy the approach into real-world applications. We propose an online monitoring service named *Behavior Manager* which monitors user's dynamic activities, and trigger a behavior check periodically. The check interval is decided by the common behavior time interval. Behavior Manager compares user's dynamic behavior with this user's history behavior, and with common behavior of the group of people who are assigned the same role, and with common behavior under the same context. An anomaly score is assigned to user's dynamic behavior by measuring the deviation between user's dynamic behavior and correspondent common behaviors. The anomaly score is categorized into quantified ranges. As a result, the dynamic behavior is marked with one of the tags: *normal*, *suspicious*, and *anomaly*.

- The analysis and algorithms introduced for constrained event clustering can be improved to determine more complex and meaningful behavior context by: i) adding more corresponding data sources into constrained event clusters to enrich attributes, such as the geographic distance between locations, user features out of event logs; and ii) applying soft constraints to collect most important events into constrained event clusters with soften restrictions. By using soft constraints to formalize desired features rather than hard constraints that cannot be violated, the significant benefit is introducing incomplete or partial behavior context information into clustering that may improve common behavior pattern extraction result.

- The proposed approach aims to discover common behavior patterns from multidimensional temporal data. The discovered patterns can be used for various purpose. Beside of anomaly detection, mining common behavior patterns are important in many application domains such as marketing and customer segmentation, user preference and habit analysis by mining common web navigation patterns, crime rate and crime model analysis by mining common criminal patterns, etc. Even though these data and applications are very diverse, and also the discovered patterns can differ largely, this generic approach is capable of guiding system administrators in learning domain-specific common behavior patterns.

- *EventGenerator* can produce large volumes of quality synthetic data that contains interesting and realistic patterns in a relatively short amount of time and at low cost. The expected time series, attribute frequencies, association and sequence patterns have been validated that they are produced correctly. Although the datasets by *EventGenerator* are produced as expected for the parameters defined, more work is required to analyze real-world data in specific domains and to develop more sophisticated models that represents real user behavior in those domains. The tool relies on some degree on randomness to select attributes and generate events therefor the result is not necessarily repeatable or deterministic.

- This thesis examined the proposed model with medium sized system (300+ users) and a medium sized dataset (300K events in 4 months). In future work, the proposed model should be examined and extended to exploring user behavior patterns in big data area. Scalability will be a bottleneck. Apache Spark [78] is a fast and general engine for large-scale data processing in distributed cluster. MLlib [78] is Apache Spark's scalable machine learning library, where association mining, sequential pattenr mining and clustering algorithms have been implemented in MLlib. We might extend the scalability of our proposed approach by migrating to Apache Spark framework for discovering user behavior patterns from big dataset.

# Appendix A

# BPQL

Extended Backus-Naur Form (EBNF) [79] notation is used for denoting the syntax of the proposed BPQL. EBNF is widely used to make formal description of the grammar of programming language. The following represents EBNF notations used in this thesis.

- Terminal identifiers/symbols are quoted '...'.
- $<$ and $>$ delimits the non-terminals.
- ::= is the definition symbol.
- { and } indicate repetition. Zero or more elements.
- | is the definition separator symbol. It separates alternatives elements.
- , is the concatenate symbol.
- ; is the termination symbol.

Table A.1 provides the keywords defined in BPQL. A programming like behavior pattern query language is defined as follows using EBNF.

Table A.1: Keywords of BPQL

| **Keywords** | **Description** |
|---|---|
| *BEGIN-BPQ* | BPQ starts. |
| *END-BPQ.* | BPQ ends. |
| *MAG* | A maximal association group. |
| *CLUSTER* | A group of related events satisfying constraints. |
| *INTRA-CONSTRAINT* | The constraints are applied on events in the same cluster. |
| *INTER-CONSTRAINT* | The constraints are applied on events from different clusters. |
| *SIZE-CONSTRAINT* | The number of events in a cluster. |

*<behavior_query> ::= BEGIN-BPQ, <cluster_specification>,END-BPQ.*

*<cluster_specification> ::= {<cluster_statement>}, <inter_cluster_constraint>;*

*<cluster_statement> ::= <cluster_name>, <source_domain>,*
*<intra_cluster_constraint>, <size_constraint>;*

*<cluster_name> ::= CLUSTER, ':=', <cluster_identifier>;*

*<source_domain> ::= MAG, ':=', {<mag_identifier>, ','},<mag_identifier>;*

*<size_constraint> ::= SIZE-CONSTRAINT, ':=', <numeric>;*

*<intra_cluster_constraint> ::= INTRA-CONSTRAINT, {<intra_constraint_expr>};*

*<inter_cluster_constraint> ::= INTER-CONSTRAINT, {<inter_constraint_expr>};*

*<intra_constraint_expr> ::= <attribute_name>, <operator>, <attribute_value>;*

*<inter_constraint_expr> ::= <cluster_identifier>, '.', <attribute_name>,*
*<operator>, <cluster_identifier>, '.', <attribute_name>;*

*<attribute_name> ::= 'User' | 'Role' | 'Location' | 'Time' | 'Service'*

*<operator> ::=* $' =' | '\neq' | '\geq' | '\leq' | '>' | '<'$

*<attribute_value> ::= <numeric>*

*<cluster_identifier> ::= <identifier>*

*<mag_identifier> ::= <identifier>*

*<identifier> ::= a string of character*

*<numeric> ::= a numeric value*

# Appendix B

# Generated Dataset with Embedded Behavior Patterns

This Appendix describes the events and user defined behavior patterns to be inserted into events. These behavior patterns are inserted into the generated events by *EventGenerator* randomly.

## B.1 Attribute Distribution

Figure B.1 presents the histograms of attribute distribution extracted from generated event dataset. Attribute *user* and *date* are randomly assigned into events, and attributes *service, action, location, time, patient* and *time* follows normal distribution with desired mean and deviation.

## B.2 Behavior Pattern Definition

Table B.1 presents the fields that are used to define user behavior pattern. Following is the configuration of user behavior patterns in JSON format.

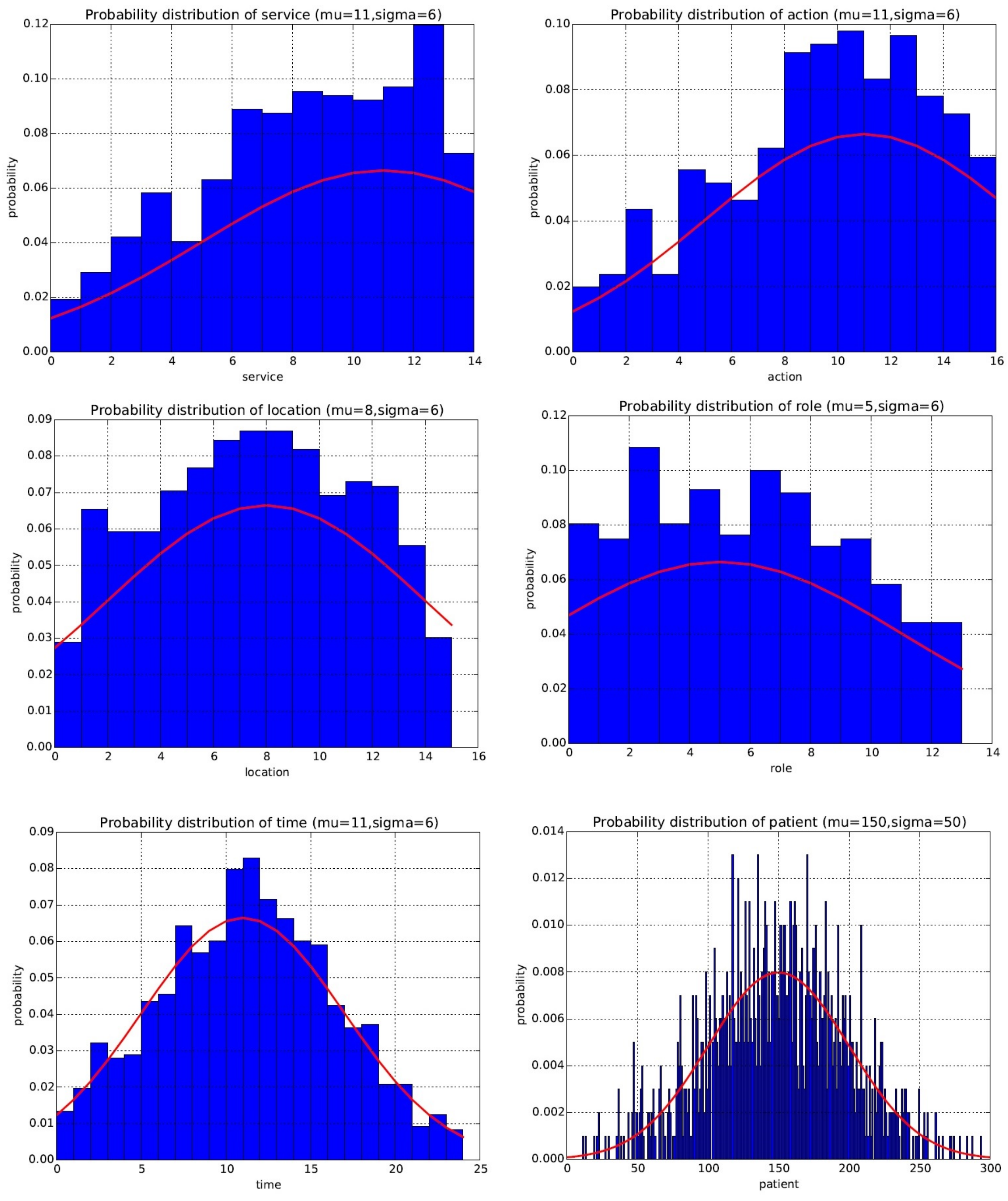


Figure B.1: Attributes with normal distribution

Table B.1: Behavior pattern definition schema

| Field | Description |
|---|---|
| id | Behavior pattern identifier |
| description | Behavior pattern description |
| actor | Behavior actor |
| context | Behavior pattern context |
| sequence | Behavior pattern sequence |
| groupSupport | Percentage of users have this behavior |
| userSupport | Percentage of specific user events contain this behavior |
| duration | Duration of behavior sequence |
| gap | Gap between successive events |

```
[
  {
    "id": "P-01",
    "description": "location pattern of ward-round",
    "actor": [
      {"role": "Nurse"}
    ],
    "context": {
      "service": "Brain and Nerves",
      "time": "10:00"
    },
    "sequence": [
      {
        "location": "Patient room #1"
      },
      {
        "location": "Patient room #3"
      },
      {
        "location": "Patient room #4"
      }
    ],
    "groupSupport": "0.2",
    "userSupport": "0.4",
    "duration": "4"
  },
  {
    "id": "P-02",
    "description": "location pattern of ward-round",
    "actor": [
      {"role": "Nurse"}
    ],
    "context": {
      "service": "Brain and Nerves",
      "time": "10:00"
    },
    "sequence": [
      {
        "location": "Patient room #2"
      },
      {
        "location": "Patient room #5"
      },
      {
        "location": "Patient room #6"
      }
    ],
    "groupSupport": "0.2",
    "userSupport": "0.4",
    "duration": "6"
  },
  {
    "id": "P-03",
    "description": "location pattern of ward-round",
    "actor": [
      {"role": "Nurse"}
    ],
    "context": {
      "service": "Brain and Nerves",
      "time": "10:00"
    },
    "sequence": [
      {
        "location": "Patient room #3"
      },
      {
        "location": "Patient room #5"
      },
      {
        "location": "Patient room #4"
      }
    ],
    "groupSupport": "0.2",
    "userSupport": "0.4",
    "duration": "10"
  },
  {
    "id": "P-04",
    "description": "Radiologist's workflow",
    "actor": [
      {"role": "Radiologist"}
    ],
    "context": {
      "service": "X-Ray",
      "service": "Lungs and Breathing"
    },
    "sequence": [
      {
        "action": "read exam",
        "gap": "2"
      },
      {
        "action": "read report",
        "gap": "2"
      },
      {
        "action": "update report"
      }
    ],
    "groupSupport": "0.25",
    "userSupport": "0.4",
    "duration": "8"
  },
  {
    "id": "P-05",
    "description": "Radiologist's workflow",
    "actor": [
      {"role": "Radiologist"}
    ],
    "context": {
      "service": "X-Ray",
      "service": "Lungs and Breathing"
    },
    "sequence": [
      {
        "action": "read exam"
      },
      {
        "action": "read order"
      },
      {
        "action": "read exam"
      }
    ],
    "groupSupport": "0.25",
    "userSupport": "0.4",
    "duration": "10"
  },
  {
    "id": "P-06",
    "description": "Radiologist's workflow",
    "actor": [
      {"role": "Radiologist"}
    ],
    "context": {
      "service": "X-Ray",
      "service": "Lungs and Breathing"
    },
    "sequence": [
      {
        "action": "read profile"
      },
      {
        "action": "read profile",
        "gap": "2"
      },
      {
        "action": "update profile"
      }
    ],
    "groupSupport": "0.25",
    "userSupport": "0.4",
    "duration": "12"
  },
  {
    "id": "P-07",
    "description": "work time sequence",
    "actor": [
      {"role": "Surgical"}
    ],
    "context": {
      "service": "Emergency",
      "service": "Blood Heart and Circulation"
    },
    "sequence": [
      {
        "time": "10:00"
      },
      {
        "time": "11:00"
      },
      {
        "time": "12:00"
      }
    ],
    "groupSupport": "0.3",
    "userSupport": "0.3"
  },
  {
    "id": "P-08",
    "description": "work time sequence",
    "actor": [
      {"role": "Surgical"}
    ],
    "context": {
      "service": "Emergency",
      "service": "Blood Heart and Circulation"
    },
    "sequence": [
      {
        "time": "10:00"
      },
      {
        "time": "12:00"
      },
      {
        "time": "13:00"
      }
    ],
    "groupSupport": "0.3",
    "userSupport": "0.3"
  },
  {
    "id": "P-09",
    "description": "work time sequence",
    "actor": [
      {"role": "Surgical"}
    ],
    "context": {
      "service": "Emergency",
      "service": "Blood Heart and Circulation"
    },
    "sequence": [
      {
        "time": "12:00"
      },
      {
        "time": "13:00"
      },
      {
        "time": "15:00"
      }
    ],
    "groupSupport": "0.3",
    "userSupport": "0.3"
  },
  {
    "id": "P-10",
    "description": "work time sequence",
    "actor": [
      {"role": "Surgical"}
    ],
    "context": {
      "service": "Emergency",
      "service": "Blood Heart and Circulation"
    },
    "sequence": [
      {
        "time": "9:00"
      },
      {
        "time": "11:00"
      },
      {
        "time": "12:00"
      }
    ],
    "groupSupport": "0.3",
    "userSupport": "0.3"
  },
]
```

## B.3 Inserted Behavior Patterns

[[*R-4*, *S-11*, *T-10*, *L-5*], [*R-4*, *S-11*, *T-10*, *L-7*], [*R-4*, *S-11*, *T-10*, *L-8*]]

[[*R-4*, *S-11*, *T-10*, *L-6*], [*R-4*, *S-11*, *T-10*, *L-9*], [*R-4*, *S-11*, *T-10*, *L-10*]]

[[*R-4*, *S-11*, *T-10*, *L-5*], [*R-4*, *S-11*, *T-10*, *L-9*], [*R-4*, *S-11*, *T-10*, *L-8*]]

[[*R-5*, *L-4*, *S-10*, *A-9*], [*R-5*, *L-4*, *S-10*, *A-10*], [*R-5*, *L-4*, *S-10*, *A-8*]]

[[*R-5*, *L-4*, *S-10*, *A-9*], [*R-5*, *L-4*, *S-10*, *A-11*], [*R-5*, *L-4*, *S-10*, *A-9*]]

[[*R-5*, *L-4*, *S-10*, *A-12*], [*R-5*, *L-4*, *S-10*, *A-12*], [*R-5*, *L-4*, *S-10*, *A-5*]]

[[*R-6*, *L-11*, *S-12*, *T-10*], [*R-6*, *L-11*, *S-12*, *T-11*], [*R-6*, *L-11*, *S-12*, *T-12*]]

[[*R-6*, *L-11*, *S-12*, *T-10*], [*R-6*, *L-11*, *S-12*, *T-12*], [*R-6*, *L-11*, *S-12*, *T-13*]]

[[*R-6*, *L-11*, *S-12*, *T-12*], [*R-6*, *L-11*, *S-12*, *T-13*], [*R-6*, *L-11*, *S-12*, *T-15*]]

[[*R-6*, *L-11*, *S-12*, *T-9*], [*R-6*, *L-11*, *S-12*, *T-11*], [*R-6*, *L-11*, *S-12*, *T-12*]]

# Appendix C

# Publications

- Knowledge-Driven User Behavior Pattern Discovery for System Security Enhancement. W. Ma, K. Sartipi, D. Bender. *International Journal of Software Engineering and Knowledge Engineering (IJSEKE)*, World Scientific Publisher. 26 pages. Volume 26, Issue 03, Page 379-405, April 2016.

- OpenID Connect as a Security Service in Cloud-based Medical Imaging Systems. W. Ma, K.Sartipi, H. Sharghi, D. Koff, P. Bak. *Journal of Medical Imaging (JMI) 3(2)*, 026501 (2016), doi: 10.1117/1.JMI.3.2.026501. SPIE Digital Library, Jun 2016.

- Security Middleware Infrastructure for Medical Imaging System Integration and Monitoring. W. Ma, K.Sartipi. *Journal of Transaction on Advanced Communications Technology (ICACT-TCAT).* Volume 4, Issue 6, Page 736-744, November 2015.

- Synthesizing Scenario-based Dataset for User Behavior Pattern Mining. W. Ma, K. Sartipi. *International Journal of Computer and Information Technology (IJCIT).* ISSN: 2279-0764, Volume 04, Issue 06, Pages 855-866. November 2015.

- Cloud-based Identity and Access Control for Diagnostic Imaging Systems.

W. Ma, K. Sartipi. *Proceedings of the International Conference on Security and Management (SAM).* The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (World-Comp), 2015 Jan 1 (p. 320).

- Federated Service-based Authentication Provisioning for Distributed Diagnostic Imaging Systems. H. Sharghi, W. Ma, K. Sartipi. *IEEE International Symposium on Computer-Based Medical Systems (CBMS 2015)*, 4 pages. June 22-25, 2015, Sao Carlos, Brazil.

- Security Middleware Infrastructure for Medical Imaging System Integration. W. Ma, K. Sartipi. H. Sharghi. *IEEE International Conference On Advanced Communication Technology (ICACT 2015)*, 5 pages, July 1-3, 2015, Seoul, Korea.

- OpenID Connect as a Security Service in Cloud-based Diagnostic Imaging Systems. W. Ma, K. Sartipi, H. Sharghi, D. Koff, P. Bak. SPIE. *Medical Imaging 2015 (International Society of Optics and Photonics).* (SPIE.Digital Library: 9 pages). Feb 21-26, 2015, Orlando, USA.

- An Agent-based Infrastructure for Secure Medical Imaging System Integration.W. Ma, K. Sartipi. *IEEE International Symposium on Computer-Based Medical Systems (CBMS 2014)*, 6 pages. New York, USA.

- An Infrastructure for Secure Sharing of Medical Images between PACS and EHR Systems. K. Sartipi, K. A. Kuriakose, and W. Ma. *IBM CASCON 2013 Conference.* November 18-20, 2013, pages 245-259, Toronto, Canada.

- Poster presentation on CASCON 2014 & CSER2014, titled "Behavior Pattern based Security Enhancement", Ma, Weina, and Kamran Sartipi.

- Poster presentation on CASCON 2013, titled "Simulation of an Infrastructure for Secure Sharing of Medical Images between PACS and EHR Systems", Sartipi, Kamran, and Krupa A. Kuriakose, and Weina Ma.

# Bibliography

[1] Market Guide for User and Entity Behavior Analytics. Technical report, Gartner Inc., 2015. https://www.gartner.com/, accessed 2016-July-29.

[2] Insider Threat Spotlight Report. Technical report, SpectorSoft Corporation., 2015. http://www.spectorsoft.com/, accessed 2016-July-29.

[3] Using Spluk UBA to Detect Insider Threats. Technical report, Splunk Inc., 2015. http://www.splunk.com/, accessed 2016-July-29.

[4] Brancik K. *Insider computer fraud: an in-depth framework for detecting and defending against insider IT attacks.* CRC Press, 2007.

[5] V. Stavrou, M. Kandias, G. Karoulas, and D. Gritzalis. Business Process Modeling for Insider threat Monitoring and Handling. In *In Trust, Privacy, and Security in Digital Business*, pages 119–131. Springer International Publishing, 2014.

[6] M. Bishop and et al. Insider Threat Identification by Process Analysis. In *Insider Threat Identification by Process Analysis. In Security and Privacy Workshops (SPW), IEEE*, pages 251–264, 2014.

[7] P. Parveen, N. Mcdaniel, Z. Weger, J. Evans, B. Thuraisingham, K. Hamlen, and L. Khan. Evolving insider threat detection stream mining perspective. *International Journal on Artificial Intelligence Tools*, 22(05), 2013.

[8] D. Agrawal, C. Budak, A. El Abbadi, T. Georgiou, and X. Yan. Big data in online social networks: user interaction analysis to model user behavior in social networks. In *In Databases in Networked Information Systems*, pages 1–16. Springer International Publishing, 2014.

[9] The Rise of User Behavior Analytics. Technical report, SpectorSoft Corporation., 2015. http://www.spectorsoft.com/, accessed 2016-July-29.

[10] There is a traitor in our midst - exploring the insider-threat market. Technical report, 451 Research LLC., 2014. https://451research.com/, accessed 2016-July-29.

[11] J.M. Spivey and J.R. Abrial. *The Z notation.* Hemel Hempstead: Prentice Hall, 1992.

[12] ISO/IEC 13568:2002. *Z formal specification notation – Syntax, type system and semantics*, 2002.

[13] D.E. Denning. An intrusion-detection model. In *Software Engineering, IEEE Transactions on*, pages 222–232, 1987.

[14] C. Phua, V. Lee, K. Smith, and R. Gayler. A comprehensive survey of data mining-based fraud detection research. arXiv preprint arXiv:1009.6119, 2010.

[15] M. H. Yarmand, K. Sartipi, and D. G. Down. Behavior-based access control for distributed healthcare systems. In *Journal of Computer Security*, pages 1–39, 2013.

[16] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. In *ACM computing surveys (CSUR), 41(3), 15*, 2009.

[17] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning.* MIT press, 2012.

[18] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: current status and future directions. In *Data Mining and Knowledge Discovery*, pages 55–87, 2007.

[19] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record. Vol. 22. No. 2. ACM*, 1993.

[20] R. Agrawal and R. Srikant. Mining sequential patterns. In *Data Engineering. IEEE*, 1995.

[21] S. Vijaylakshmi, V. Mohan, and S. Suresh Raja. Mining of users access behavior for frequent sequential pattern from web logs. In *International Journal of Database Management System (IJDM)*, 2010.

[22] B. Livshits and T. Zimmermann. DynaMine: finding common error patterns by mining software revision histories. *ACM SIGSOFT Software Engineering Notes*, 5(1):296–305, 2005.

[23] M. N. Garofalakis, R. Rastogi, and K. Shim. SPIRIT: Sequential pattern mining with regular expression constraints. In *VLDB. Vol. 99*, 1999.

[24] C.C. Yu and Chen. Y.L. Mining sequential patterns from multidimensional sequence data. In *Knowledge and Data Engineering, IEEE Transactions*, pages 136–140, 2005.

[25] J. Stefanowski and R. Ziembinski. Mining context based sequential patterns. In *Advances in Web Intelligence. Springer Berlin Heidelberg*, pages 401–407, 2005.

[26] H. Mannila, H. Toivonen, and A.I. Verkamo. Discovery of frequent episodes in event sequences. In *Data Mining and Knowledge Discovery*, pages 259–289, 1997.

[27] Z. He, X. Xu, J.Z. Huang, and S. Deng. FP-outlier: Frequent pattern based outlier detection. In *Comput. Sci. Inf. Syst 2(1)*, pages 103–118, 2005.

[28] M. Gupta, J. Gao, and J. Han. Community trend outlier detection using soft temporal pattern mining. In *In Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 692–708. Springer Berlin Heidelberg, 2012.

[29] S. Xie, G. Wang, S. Lin, and PS. Yu. Review spam detection via temporal pattern discovery. In *InProceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 823–831. ACM, 2012.

[30] D. Brauckhoff, X. Dimitropoulos, A. Wagner, and K. Salamatian. Anomaly extraction in backbone networks using association rules. In *IEEE/ACM Transactions on Networking (TON)*, pages 1788–1799, 2012.

[31] C.C. Aggarwal and C.K. Reddy. *Data clustering: algorithms and applications.* CRC Press, 2013.

[32] T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):881–892, 2002.

[33] H.P. Kriegel and et al. Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(3):231–240, 2011.

[34] R. Xu and D. Wunsch. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, 2005.

[35] S. Basu, A. Banerjee, and R.J. Mooney. Active Semi-Supervision for Pairwise Constrained Clustering. *SDM*, 4:333–344, 2004.

[36] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrodl. Constrained k-means clustering with background knowledge. *ICML*, 1:577–584, 2001.

[37] K.L Wagstaff. *Intelligent clustering with instance-level constraints*. PhD thesis, Cornell University, 2002.

[38] TW. Liao. Clustering of time series dataâĂŤa survey. In *Pattern recognition*, pages 1857–1874, 2005.

[39] L. Bergroth, H. Hakonen, and T. Raita. A survey of longest common subsequence algorithms. In *String Processing and Information Retrieval (SPIRE), IEEE*, pages 39–48, 2000.

[40] E.W. Myers. AnO (ND) difference algorithm and its variations. *Algorithmica*, 1(1-4):251–266, 1986.

[41] L. Bergroth, H. Hakonen, and T. Raita. A survey of longest common subsequence algorithms. In *String Processing and Information Retrieval*, pages 39–48, 2000.

[42] A. Gabadinho, G. Ritschard, M. Studer, and N.S. MÃijller. Extracting and rendering representative sequences. In *Knowledge Discovery, Knowlege Engineering and Knowledge Management*, pages 94–106, 2011.

[43] Fokkink W. Introduction to process algebra. In *Springer Science and Business Media*, 2013.

[44] A. Dogac, L. Kalinichenko, T. ÃŰzsu, and A. Sheth. Workflow management systems and interoperability. In *Springer Science and Business Media*, 2012.

[45] The R project for statistical computing website. http://www.r-project.org/, accessed 2016-July-29.

[46] CRAN Task Views of R-package. https://cran.r-project.org/web/views/, accessed 2016-July-29.

[47] R-package arules: Mining Association Rules and Frequent Itemsets. http://cran.r-project.org/web/packages/arules/index.html, accessed 2016-July-29.

[48] R-package arulesSequences: Mining frequent sequences. http://cran.r-project.org/web/packages/arulesSequences/index.html, accessed 2016-July-29.

[49] M. Maechler, P. Rousseeuw, A. Struyf, Hornik K. Hubert, M., M. Studer, and P. Roudier. cluster: "Finding Groups in Data", 2015. R package version 2.0.3.

[50] A. Gabadinho, G. Ritschard, N.S. Mueller, and M. Studer. Analyzing and visualizing state sequences in R with TraMineR. *Journal of Statistical Software*, 40(4):1–37, 2011.

[51] W. Ma and K. Sartipi. An Agent-Based Infrastructure for Secure Medical Imaging System Integration. In *IEEE 27th International Symposium on Computer-Based Medical Systems (CBMS)*, pages 72–77, 2014.

[52] K. Sartipi, K. Kuriakose, and W. Ma. An Infrastructure for Secure Sharing of Medical Images between PACS and EHR Systems. In *International Conference on Computer Science and Software Engineering (CASCON)*, pages 245–259, 2013.

[53] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C.W. Wu, and V.S. Tseng. SPMF: a Java open-source pattern mining library. *The Journal of Machine Learning Research*, 15(1):3389–3393, 2014.

[54] M. Hahsler, C. Buchta, B. Gruen, and K. Hornik. arules: Mining Association Rules and Frequent Itemsets, 2015. R package version 1.3-1.

[55] I. Sommerville and et al. *Software Engineering: Chapter 27 Formal Specification.* Pearson, 2009.

[56] I. Sommerville and et al. *Software Engineering: Chapter 11 Model-based Specification.* Pearson, 1995.

[57] M. A. Whiting, J. Haack, and C. Varley. Creating realistic, scenario-based synthetic data for test and evaluation of information analytics software. In *Proceedings of the 2008 Workshop on BEyond time and errors: novel evaLuation methods for Information Visualization. ACM*, 2008.

[58] Market-Basket Synthetic Data Generator, 2011. https://synthdatagen.codeplex.com/, accessed 2016-July-29.

[59] An Open-Source Data Mining Library, 2015. http://www.philippe-fournier-viger.com/spmf/, accessed 2016-July-29.

[60] Y. Pei and O. Zaiane. A synthetic data generator for clustering and outlier analysis. In *Department of Computing science, University of Alberta, edmonton, AB, Canada*, 2006.

[61] Sharghi H., Ma W., and Sartipi K. Federated Service-based Authentication Provisioning for Distributed Diagnostic Imaging Systems. In *IEEE International Symposium on Computer-Based Medical Systems (CBMS)*, 2015.

[62] Yarazavi A. and Sartipi K. Consultant-as-a-service: an interactive and context-driven approach to mobile decision support services. In *International Conference on Computer Science and Software Engineering (CASCON)*, pages 274–282, 2013.

[63] J. de Ona, G. Lopez, and J. Abellan. Extracting decision rules from police accident reports through decision trees. In *Accident Analysis and Prevention, 50*, pages 1151–1160, 2013.

[64] W. J. Dixon and F. J. Massey. *Introduction to statistical analysis.* New York: McGraw-Hill., 1969.

[65] Introducing JSON website. http://json.org/, accessed 2016-July-29.

[66] RFC 3881: Security Audit and Access Accountability Message XML Data Definition for Healthcare Applications. https://tools.ietf.org/html/rfc3881, accessed 2016-July-29.

[67] Python programming language website. https://www.python.org/, accessed 2016-July-29.

[68] EventGenerator at GitHub. https://github.com/maweina/EventGenerator, accessed 2016-July-29.

[69] MARC-HI Everest Framework. http://te.marc-hi.ca/, accessed 2016-July-29.

[70] B. Liu, H. Wynne, and Y. Ma. Mining association rules with multiple minimum supports. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, 1999.

[71] Gephi - The Open Graph Viz Platform. http://gephi.github.io/, accessed 2016-July-29.

[72] C.C. Aggarwal and C.K. Reddy. *Data clustering: algorithms and applications.*, volume 29. 2000.

[73] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.C. Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern

growth. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE Computer Society, 2001.

[74] AWS CloudTrail. https://aws.amazon.com/cloudtrail/, accessed 2016-July-29.

[75] P. Roudier. Agglomerative Nesting (Hierarchical Clustering). https://stat.ethz.ch/R-manual/R-devel/library/cluster/html/agnes.html, accessed 2016-July-29.

[76] Cut a Tree into Groups of Data. https://stat.ethz.ch/R-manual/R-devel/library/stats/html/cutree.html, accessed 2016-July-29.

[77] J. Davis and M. Goadrich. The relationship between Precision-Recall and ROC curves. In *23rd international conference on Machine learning, ACM*, pages 233–240, 2006.

[78] Apache Spark Website. http://spark.apache.org/, accessed 2016-July-29.

[79] L.M Garshol. BNF and EBNF: What are they and how do they work. In *acedida pela Ãžltima vez em, 16*, 2003.