International Journal of Software Engineering and Knowledge Engineering

Knowledge-driven User Behavior Pattern Discovery for System Security Enhancement

Weina Ma
Department of Electrical, Computer, and Software Engineering
University of Ontario Institute of Technology
Oshawa, Ontario L1H 7K4,Canada
Weina.Ma@uoit.ca

Kamran Saritpi
Department of Electrical, Computer, and Software Engineering
University of Ontario Institute of Technology
Oshawa, Ontario L1H 7K4,Canada
Kamran.Sartipi@uoit.ca

Duane Bender
Department of Electrical and Computer Engineering Technology
Mohawk College
Hamilton, Ontario L8N 3T2, Canada
Duane.Bender@mohawkcollege.ca

Insider threads posed by authorized users have caused significant security and privacy risks to IT systems. The behavior of authorized users in using system services must be monitored and controlled. However, the administrators in large distributed systems are overwhelmed by the number of system users, the complexity and changing nature of user activities. This paper presents a new generation of intelligent decision support system that effectively assists system administrators to get deep insight into the system user's dynamic behavior patterns. With these patterns, the system administrators are capable of constructing dynamic refinement to the existing security policies. We explore the method of interactively and incrementally extracting user's behavior patterns by combining data mining techniques with domain and system knowledge, and applying such knowledge to provide recommendations throughout the whole process. A prototype tool has been developed to analyze the audit logs from distributed medical imaging systems to validate the proposed approach.

Keywords: Behavior pattern discovery; association mining; sequential pattern mining; pattern query language; security enhancement.

1. Introduction

Despite the availability of common security mechanisms such as authentication, authorization and secure communication in most systems, authorized users intentionally or carelessly demonstrate risky behaviors that may cause data leakage or damage to the protected resources. Behavioral activities of authorized users must

be monitored and controlled to protect user's private data and identify malicious behaviors. It has been a big challenge for system administrators in large distributed systems to detect user's unusual behavior, due to: the large number of system users whose behaviors in using the system services must be monitored and controlled; and the complexity and changing nature of user behavioral patterns.

This paper introduces a knowledge-driven user behavior-pattern discovery approach by analyzing system access-logs, with a step-by-step guidance for system administrators throughout the whole process. We propose a generic behavior model that allows the user 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).

Sequential pattern mining is a data mining technique that can be used to discover the time-based correlation relations that exist among customer transaction history. It identifies the frequent sequential patterns that occur in more than a user specified number (i.e., minimum support threshold) in a sequence database. Where the support of a sequential pattern is the number of sequences in a sequence database in which the pattern exists. In our approach, 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 obtain user's frequent action sequences. The task of discovering frequent sequential patterns in access logs generated from large distributed systems is challenging, because the algorithm needs to process an explosive number of possible sequences.

To reduce the search space for sequential pattern mining without significant information loss, we applied association mining operation as pre-processing of event database to discover highly related event groups. Association mining is a process of discovering highly related events from database according to a user defined minimum support threshold. In our approach, the database is a set of user access events, and each event consists of a set of attributes. Based on the highly associated groups of events acquired by association mining process, we propose an association-based similarity metric to measure the similarity between events. We create a database of seed-domains, where each event has a corresponding seed-domain and then assign events that have non-zero similarity values with that event into its seed-domain. The seed-domains are then ranked based on their statistical data such as the number of events in the domain, and the average association value in the domain.

Many algorithms have been designed for pattern mining but few of the discovered patterns have semantic meaning or particular interest to business. In our approach, having the ranking list of event domains and the shared attributes in the association groups, the system administrators are able to compose complex behavior pattern queries to express their interests and focus. We designed a behavior pattern query language capable of incorporating knowledge obtained from system analysis and user's experience, which results in discovering more valuable user behavior patterns with limited search space.

As a case study, we developed a prototype for analyzing audit logs from a distributed medical imaging system. The experimentations indicate that the proposed approach effectively guides and assists users in discovering interesting behavior patterns through providing recommendations in the whole process.

The contribution of this paper can be summarized as follows: i) providing a user behavior pattern discovery environment that can be applied on large distributed systems; ii) advancing knowledge-driven approaches and processes for guiding system administrators to explore the existing behavior patterns for administration purpose; iii) proposing user behavior-oriented modeling and behavior pattern query language that allows the analyst to describe a complex behavior pattern to be discovered by the pattern mining engine; and iv) presenting an association-based similarity metrics that measures the significance of events based on data mining techniques.

The remaining of this paper is organized as followings. Related work is discussed in Section 2, and the relevant background knowledge for this paper is presented in Section 3. In Section 4 our knowledge-driven behavior pattern discovery approach is explained. Section 5 is allocated to the case study. Finally, discussion and conclusion are presented in Sections 6 and 7.

2. Related Work

In this section we discuss the approaches that are related to our work.

Intrusion Detection. Intrusion detection is the process of monitoring network or system activities for identifying and responding to malicious activities. Behavior-based approaches using data mining technology have been widely studied both in network based intrusion detection and host based intrusion detection. Comparatively, application level intrusion detection or user-centric behavior analysis are less investigated. Stiawan et al. [1] proposed behavior-based prevention to trigger mechanism and analyze correlation outbound traffic based on habitual activity from inside user. Shabtai et al. [2] presented a malware detection framework for Android mobile devices which employs machine learning technologies. This paper evaluated several abnormal detection/classification algorithms and feature selection methods in order to detect new malware based on sample of known malwares. Ye et al. [3] proposed an intrusion detection approach based on system call sequences and rules extraction, which constructs the normal behavior model in terms of the system call sequences generated during the normal execution of a process and detects the behavior that deviates from the model. Our research is an attempt to explore unknown user-centric behavior patterns through knowledge-driven data mining techniques.

Behavior Analysis. Behavior has been increasingly recognized as a key research object in business and security area. Several behavior-based analysis applications have been studied in recent years. Lin [4] designed an anti-fraud system for online transaction with credit card. The proposed system constitutes a behavior analysis module to analyze the historical consumption records which is used to

judge the current trading risk level. Guan et al. [5] did statistical analysis on user behavior of micro-blogging website during hot social events. Among the 21 selected events, they found some interesting conclusions through analyzing user's posting and reposting characteristics. Cinque et al. [6] presented a rule-based approach to analyze software failure behavior through mining event logs. Many existing behavioral analysis systems typically operate by injecting known patterns such as rules, transactions and sequences to unknown dataset to match with similar new instances. This research explores the unknown user behavior patterns without priori knowledge.

Behavior-based Access Control. Behavior-based access control for distributed healthcare systems is initially proposed by Yarmand and Sartipi [7]. The proposed access control model captures the dynamic behavior of the user, and determines access rights through comparing with the expected behavior. Ideally, the distance between observed behavior and expected behavior is significant if the user acts abnormally. This model is also applied in the security sharing of medical images through action-based access control mechanism and user-behavior based policy enhancement procedure [8]. However, the method of acquiring decent expected behavior which is crucially important is still an open issue.

3. Background

In this section, we provide an overview of different background knowledge required for our approach.

3.1. *Frequent Pattern Mining*

Frequent patterns include itemsets, subsequences and substructures that appear in a dataset with frequency no less than a user specified threshold [9]. 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 patterns. 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.

3.2. *Association Mining*

The concept of association mining was first introduced by Agrawal [10] 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 is a set of items

with the cardinality of k which is called *k-itemset*. The support of an itemset is the number of transactions (i.e., baskets) that contain that itemset in the transaction database. The Apriori algorithm [10] 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.

3.3. *Sequential Pattern Mining*

Agrawal [11] 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 *customoer-sequence*. Mining sequential patterns is the process of finding the frequent sub-sequences 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 that sequence as a sub-sequence. The sequential pattern mining algorithm *AprioriAll* [11] 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. A number of applications utilize sequenced data, including: customer shopping sequences, web click streams, and biological sequences. 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 [12], multi-dimensional sequential pattern mining [13], context-based sequential pattern mining [14], and frequent

episode discovery in event sequences [15].

3.4. *Knowledge-driven Decision Support System*

A knowledge-driven decision support system (KD-DSS) provides specialized problem-solving expertise stored as facts, rules, procedures, or in similar structures [16]. It is an interactive software-based system intended to suggest or recommend actions to decision makers using a knowledge base. The knowledge discovery process may consist of following steps: i) data integration and feature selection; ii) data mining to discover and gain knowledge; iii) knowledge interpretation and representation. The knowledge is explicitly represented via automatic tools as ontology or rules which assist in DSS behave like an intelligent consultant: supporting decision makers by gathering and analyzing evidence, identifying and diagnosing problems, proposing possible actions, and evaluating the proposed actions. KD-DSS has the capabilities of self-learning, identifying the associations between raw data, integrating with data mining techniques to discover hidden patterns, and performing heuristic optimizations [17]. These abilities turn KD-DSS into an intelligent process which improves the accuracy of decision making.

4. Approach

In this section, we propose an approach to discover and analyze access behaviors of authorized users in large distributed systems. In Subsection 4.1, we present an example to illustrate the maximal association relationship and the behavior pattern mining technique in our approach. Subsection 4.2 defines an abstract behavior model and the basic concepts that are employed in the proposed model. In Subsection 4.3, the behavior pattern discovery process and technologies are introduced. Subsection 4.4 illustrates the proposed behavior pattern query language that allows system administrators to compose complex user behavior patterns.

4.1. *Motivating Example*

Table 1 demonstrates an example event database from medical imaging systems that trace three users' accesses 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 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 one new image of the patient; ii) he reviews patient's history examinations; and iii) he creates a diagnosis report for the new examination.

Table 1. An example of event database

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

Table 2. The 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) Afternoon(12:00-24:00)	T-1 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, Office	L-1, L-2 L-3
Operation	Order an Exam, Search & View, Take Exam, Read, Create	O-1, O-2, O-3 O-4, O-5
Resource Type	Image, History Exams, Diagnose Report, Demographic Data	S-1, S-2 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

Table 3. Transformed event database used in mining

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-2	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

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?

Event Representation

In our approach, each event is represented as a group of attributes. Attribute names, values and encoding used in the example are shown in Table 2. The encoded event database used in mining is shown in Table 3.

Maximal Association

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. We refer to the group of events as the "*baskets*" and the shared set of attribute values as the "*itemset*" (each attribute value is viewed as an item). In this sense, the whole group of baskets and their itemset are denoted as a maximal association group (MAG). We aim to

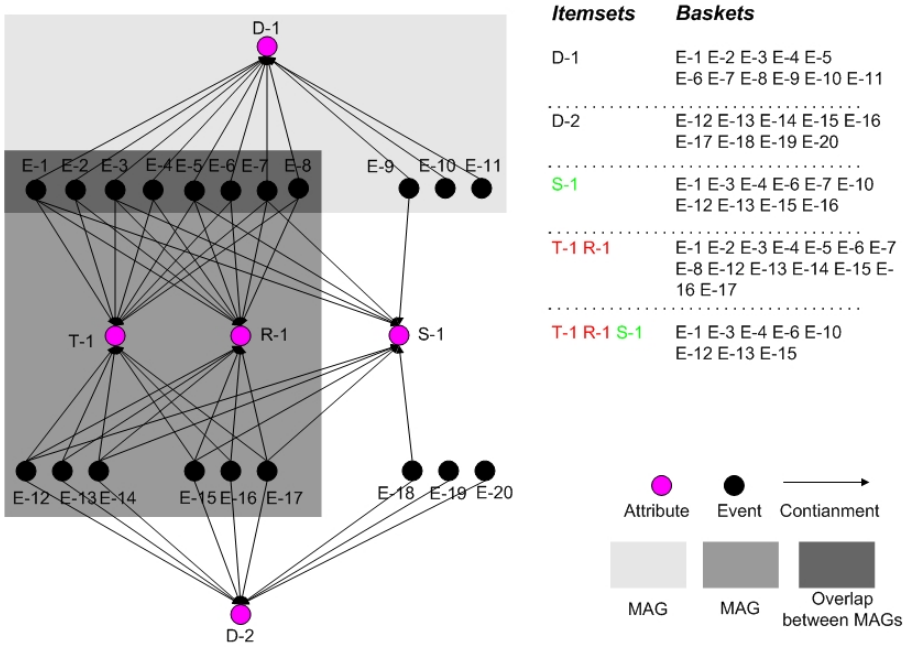

Fig. 1. Maximal association groups extracted from encoded event database

find large MAGs whose both baskets and itemsets are large. Association mining algorithm *Apriori* [10] (defined in Subsection 3.2) is applied on the encoded event database with minimum support 8, and the algorithm extracts 9 frequent itemsets. We remove the MAGs whose itemsets are subsets of the final MAGs. This phase deletes a large number of redundant MAGs. Figure 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 example, we can see from Figure 1 that: events {E-1, E-2, E-3, E-4, E-5, E-6, E-7, E-8} exist in two MAGs; and attributes {T-1, R-1} and {S-1} appear in itemsets of two MAGs.

The association-based 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 as follows:

$$sim(e_i, e_j) = max_{g_x} \ (|itemset(g_x)| + w * |baskets(g_x)|)$$

where $0<w<1$ is the weight of the shared attributes compared with the sharing events. The event association is considered as a measure of similarity between two events that allows to identity the events of a group of highly related. In general, the number of shared attributes contributes more on the closeness of the events than

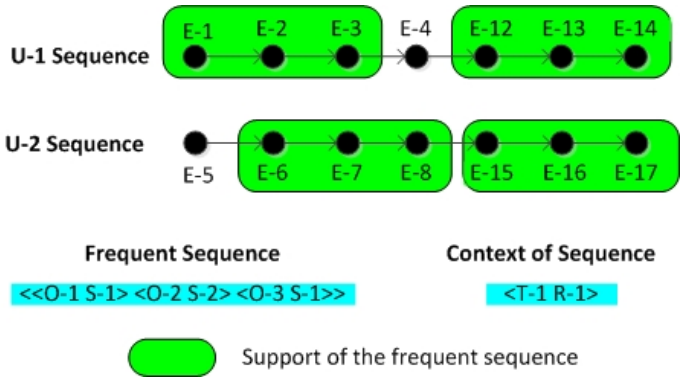

Fig. 2. Extracted behavior pattern of radiology-workflow

the number of sharing events, if a group of events are considered for their similarity. We use a value of w=0.5 in this example. The ranking result of event similarities provides user a clue to the significance of events and the common characteristics among such events. As shown in Figure 1, the MAG with itemset {T-1, R-1} received the highest similarity value "9" since its itemset length is "2" and the size of baskets is "14". This group of events is collected for subsequent behavior pattern mining phase.

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 events with rich associations. To reduce the search space in subsequent behavior mining phase, we propose a behavior pattern query language that allows users to express their interest. The behavior pattern query language will be discussed in Subsection 4.4.

Behavior Pattern Mining

In order to mine frequent behavior patterns under specific context, we apply sequential pattern mining (defined in Subsection 3.3) on the groups of associated-events 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 2 shows the extracted behavior patterns from associated event-group with common

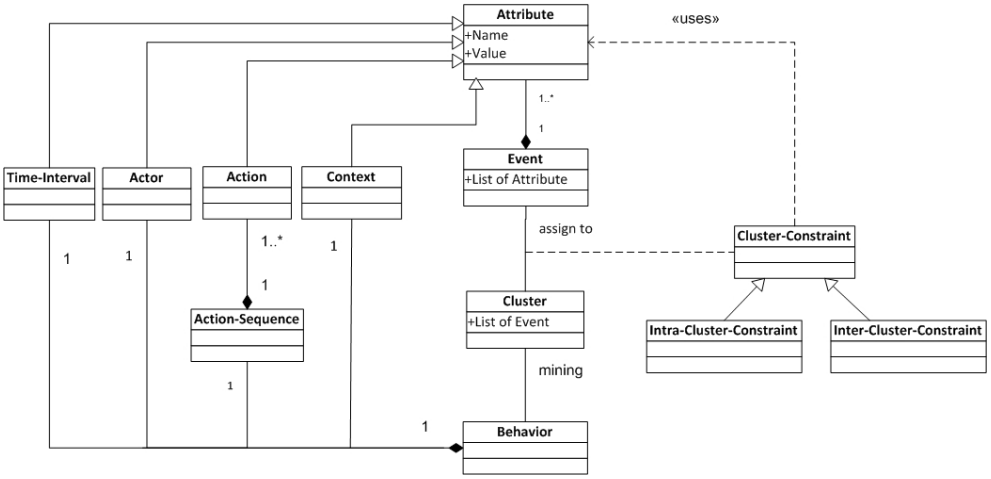

Fig. 3. Class diagram of an abstract user behavior domain model.

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 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 appearances of such action-sequence in user-sequences. So the support of the action sequence in our example is 4 (2 for U-1 and 2 for U-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.

4.2. *Abstract Behavior Model*

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 (for simplicity, we refer to attribute-value and attribute-values as "attributeV" and "attributeVs"). Whenever any attributeV of the event changes, a new event is recorded.

User behavior (for simplicity, we refer to it as just "Behavior") is extracted from a set of user-system interactions. *Behavior* is defined as: *consistent observations of a sequence of actions performed by the same user, under certain environment as well as during a specified time interval*. Behavior is represented as a quadruple:

Behavior = <Actor, Action Sequence, Context, Time Interval>

where i) *Actor* issues a behavior; ii) *Action Sequence* is the sequence of actions

performed by the Actor; iii) *Context* is the circumstances in which a behavior takes place; and iv) *Time Interval* is the time duration within which the behavior is recovered.

Figure 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. Class *Cluster* is an aggregation of class *Event*. An *Event* will be assigned to a *Cluster* if it satisfies the *Cluster-Constraints*. *Cluster-Constraint* is a class which restricts the cluster size to enhance the similarity among events in the cluster. *Cluster-Constraint* is defined by the user using important attributes and it is either an *Intra-Cluster-Constraint* (defined for one cluster) or *Inter-Cluster-Constraint* (defined between two clusters). Class *Behavior* contains some common characteristics among events that are mined from cluster through sequential pattern mining. The classes *Actor, Action-Sequence, Context* and *Time-Interval* that inherit from class attribute are designed to describe the four intergradient of a behavior. Class *Actor* issues a behavior; an actor can be an individual person or a group of people. Class *Action* represents the operation that an actor performs on the system resources in a single user-system interaction (class *Event* records all aspects of a user-system interaction). Class *Action-Sequence* represents the order of actions that the actor performs during the specified time-interval. Class *Context* indicates the environment in which a behavior always happens. Class *Time-Interval* describes it is a user's daily behavior, or weekly behavior, etc.

4.3. *Proposed Framework and Process*

Figure 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 a maximum set of events that all share a maximum set of attribute values. 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 two events, 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. Also, in order to restrict the search spaces during populating the clusters, we create a database of seed-domains, in which for each event (as a potential seed of a cluster) all other events that have non-zero similarity values with the seed-event will be collected as the domain of the seed-event. This database will be used for two purposes: i) for ranking seed-events based on different domain statistics to provide important

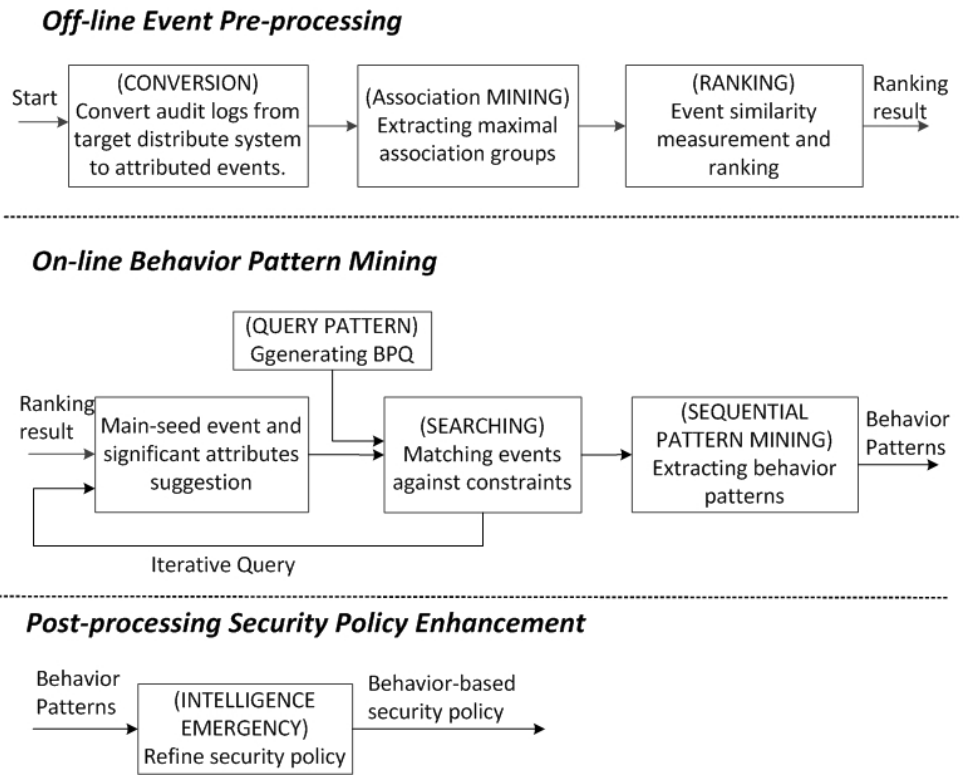

Fig. 4. The proposed behavior pattern recovery process.

recommendations for the use; and ii) as restricted search spaces to collect events satisfying user defined constraints in online behavior pattern mining phase. The ranking is based on a weighted average of criteria, such as: the size of the domain, the average similarity value among all events inside the domain, and the maximum similarity that exists in the domain. For each ranked domain our approach provides seed event along with the shared attributes that exist in the larges MAG in that domain, which allows the user to focus on the significant attributes in composing the behavior pattern query.

Step 2, On-line Behavior Pattern Mining. Using an iterative process, the user incrementally selects a main-seed event from the suggested main-seed list; and generates a BPQ (Behavior Pattern Query) based on the suggested attributes and user's domain knowledge to collect highly related events from the seed-domain. A link-constraint is defined using attributeVs 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 attributeVs that the user is interested to investigate. The inter-constraints allow separating the clusters from each others; then the user

can convert the inter-constraints into intra-constraints during the searching operation. The Sequential pattern mining is applied on each constrained-cluster to extract a number of frequent action-sequences. A common action-sequence within the event-sequence of the same user is that user's behavior. If a user's common action-sequence is also seen in other users' event 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 frequent action-sequences extracted from this cluster. Through analyzing the time interval of each frequent action-sequence, all such frequent action-sequences are categorized into quantified ranges such as hourly, daily and weekly. Finally a number of common behavior patterns are extracted and represented as the quadruple: actor, action-sequence, context and time interval.

Step 3 Post-processing for Security Policy Enhancement. The result of the recovered behavior patterns are visualized and displayed in a user-friendly manner. The user investigates the properties of the recovered behaviors, such as the actors who issued the behavior, the corresponding actions of the behavior, the context under which the behavior was performed, and the time window of the behaviors. Based on the characteristics of the recovered similar/outlier behavior patterns, there are huge possibilities to identify anomaly behavior of some users and to profile the expected behavior. Finally, the system administrators are capable of refining the existing security polices through comparing and analyzing the distances between the knowledge acquired from the recovered behavior patterns and the existing security policies.

4.4. *Behavior Pattern Query Language (BPQL)*

The proposed BPQL describes the high level and abstract clusters and their intra-cluster/inter-cluster constraints that allow the analysts (administrators) to identify the system-user's behavior patterns that match with the defined high level constraints. Appendix A and Appendix B provide the keywords and syntax for the proposed BPQ. The followings are two BPQL query examples.

- BPQL query 1: Explore the behavior during rush hour.

Below is an example of exploring behavior patterns during rush hour at different locations. For example, event E-1 and event E-2 are on the top of the ranking list generated by the seed-domain selection process in the pre-processing phase. The shared attributes in the MAG containing E-1 is rush hour (from 9:00am to 11:00am) and location (city Oshawa); the shared attributes in the MAG containing E-2 is rush hour (from 9:00am to 11:00 am). The analyst with domain knowledge may wish to investigate the people who have more than 10 access requests within 2 hours. According to system recommendations and user's domain knowledge, the analyst selects event E-1 as the main-seed of cluster C-1, and E-2 as the main-seed of cluster C-2. Constrained-cluster C-1 includes the events issued by the users who

access patient's examinations more than 10 times each day during rush hour at city Oshawa between Jan 01, 2014 and Feb 01, 2014; and constrained-cluster C-2 collects the events issued by the users who request to access patient's examinations more than 10 times each day during rush hour at a location where the distance is between 50km and 100km from Oshawa from Jan 01, 2014 to Feb 01, 2014

BEGIN-BPQL
CLUSTER: C-1
MAIN-SEED: EVENT E-1
INTRA-CONSTRAINT:
Location = 'Oshawa';
Data_type = 'Examination';
Date > 2014-01-01; Date < 2014-02-01;
Time > 9:00; Time < 11:00;
GROUP BY User, Date:
HAVING COUNT > 10;
CLUSTER: C-2
MAIN-SEED: EVENT E-2
INTRA-CONSTRAINT:
Date_type = 'Examination';
Date > 2014-01-01; Date < 2014-02-01;
Time > 9:00; Time <11:00;
GROUP BY User, Date:
HAVING COUNT > 10;
INTER-CONSTRAINT: CLUSTER C-1, CLUSTER C-2
Location > 50km; Location < 100km;
END-BPQL

A number of frequent behavior patterns may be discovered through applying sequential pattern mining on each constrained-cluster. Then the analyst investigates the discovered behavior patterns to explore user's daily behavior patterns during rush hour such as: i) who is the most active user during the rush hour; ii) are the most active users same at different locations; iii) what is the frequent action-sequence during rush hour; iv) what is the similar frequent action-sequence between city Oshawa and the locations that is near Oshawa.

- BPQL query 2: Discover abnormal behavior.

The example shown below is intended to detect abnormal user behavior patterns. For example, E-1 and E-2 are the top 2 events generated by seed-domain selection process in the pre-processing phase. The constraints imposed on the domain E-1 are location (city Oshawa) and action (login); the constraint imposed on domain E-2 is action (login). Based on such recommendations, the user selects E-1 as the main-seed of cluster C-1 and E-2 as the main-seed of cluster C-2. Constrained-cluster C-1

collects user login events at city Oshawa, and constrained-cluster C-2 gathers user login events at a location with more than 100km from Oshawa.

```
BEGIN-BPQL
    CLUSTER: C-1
        MAIN-SEED: EVENT E-1
        INTRA-CONSTRAINT:
            Location = 'Oshawa';
            Login = 'True';
    CLUSTER: C-2
        MAIN-SEED: EVENT E-2
        INTRA-CONSTRAINT:
            Login = 'True';
    INTER-CONSTRAINT: CLUSTER C-1, CLUSTER C-2
        Location > 100km;
END-BPQL
```

These two constrained-clusters allow the analyst to explore the abnormal behavior patterns such that the same user login at Oshawa followed by login at location more than 100km from Oshawa within 10 minutes. However, 100km is impossible to be reached in 10 minutes.

5. Case Study

To demonstrate the functionality and performance of the proposed approach to knowledge-driven user behavior-pattern discovery, we developed a prototype and applied on the audit logs from a distributed medical imaging system. The audit events are collected from a distributed PACS (Picture Archiving and Communication System) and IHE (Integrating the Healthcare Enterprise) integrated imaging systems using MARC-HI Everest Framework [18]. 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" [19] 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.

We developed an audit log analyzer according to the schema defined in RFC3881 to convert the collected audit logs into attributed events. In total we collected 695 user access events from the distributed medical imaging system running at Mohawk College within a month. Table 4 indicates: attribute names; mapped attributes onto the XML schema of RFC3881; domains of attribute values; and selected minimum

Table 4. The attributes converted from audit logs of distributed medical imaging system

Attribute Name	**Data Definition in XML schema**	**Attribute Domain**	**MinSup**
User	ActiveParticipant/UserIsRequestor='true'/UserID	329	0.002
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	0.002
Resource	ParticipantObjectIdentification/ ParticipantObjectTypeCode='2'/ParticipantObjectID	492	0.005
Date	EventIdentification/EventDateTime	25	0.02
Time	EventIdentification/EventDateTime	24	0.04

supports for association mining which will be further discussed in the following subsection.

5.1. *Maximal Association*

Since Apiori algorithm [10] considers only one minimum support value (*minsup*) for the whole dataset, the model 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. 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 the *minsup* is set too high, the patterns that involve rare attribute values will be deleted. On the other hand, if the *minsup* is set too low, it may cause explosion of discovered patterns.

The method of mining association rules with multiple *minsup* is introduced by Liu [20]. The approach allows the user to define *minsup* for each attributeV, while we define *minsup* for each category of attributes. The value of *minsup* of an attribute category is selected as the average probability of an attributeV 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 attributeV of "Role" is 0.25. The selected *minsup* of each attribute is presented in Table 4. 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, the *minsup* of attribute "Patient" is very low (0.002) that only a small group of events are related to accessing the records of the same patient. 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, P-2" is frequent if its support is

Fig. 5. Visualization of the association-based similarity among events.

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.

The **similarity** values among the events are calculated based on the method defined in Subsection 4.1, which utilizes the extracted maximal association groups. Figure 5 illustrates the undirected graph representation of the complete event set in our case study. We used Gephi [21] 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 several nodes, its color becomes heavier. Gephi also provides some layout algorithms that push out unconnected nodes far away from the rest of the graph, and provides modularity statistic algorithms that allows for clustering in the graph by identifying components that are highly interconnected. After applying layout and modularity statistic algorithms, we obtained the view of Figure 5 which shows several clusters. However, this automatic tool does not provide any information about the common characteristics of the clusters.

5.2. *Constrained-clusters*

We create 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 assign 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 collect 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 Subsection 4.4 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.

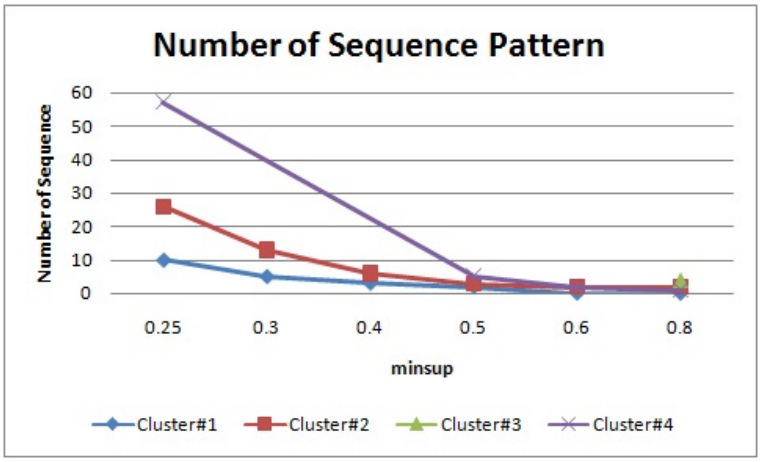

Fig. 6. The number of discovered frequent sequence patterns in each cluster with different minsup.

5.3. *Behavior Pattern Mining*

Sequential pattern mining algorithm Apriori [11] (defined in Subsection 3.3) 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, where each cluster is obtained using applying association mining operation and relative *minsup* values from 0.25 to 0.8. Figure 6 shows the number of discovered frequent sequence patterns for 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 7 and 8). 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 7 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 7 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

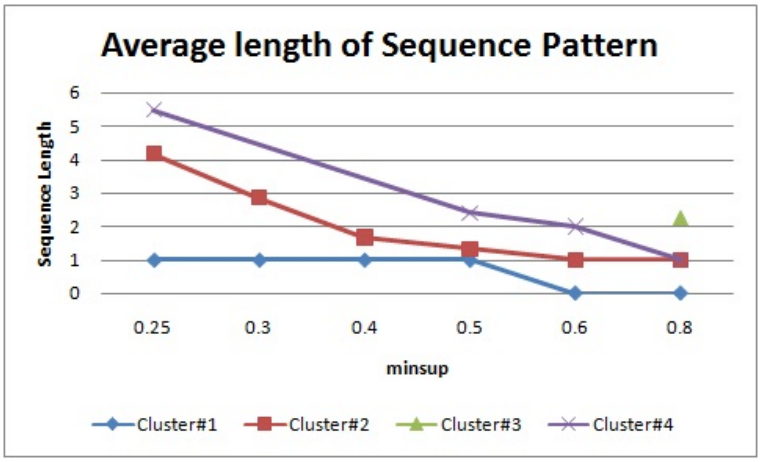

Fig. 7. The average length of discovered frequent sequence patterns in each cluster with different minsup.

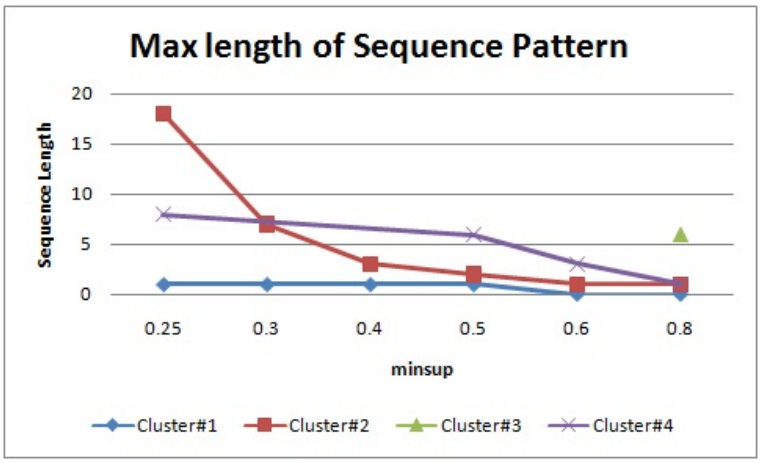

Fig. 8. The maximum length of discovered frequent sequence patterns in each cluster with different minsup.

helps to discover long length sequence patterns.

Figure 8 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

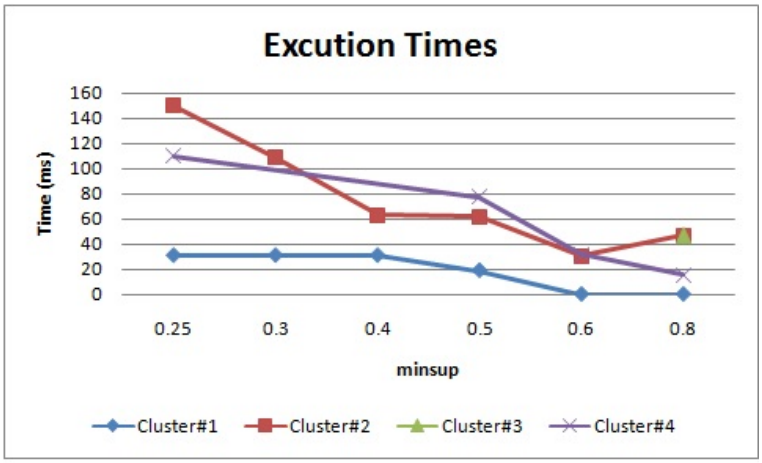

Fig. 9. The execution time of mining each cluster with different minsup.

knowledge-driven approach, maximal association and user produced cluster constraints based on recommendations, improves the discovery of behavior patterns.

Figure 9 shows the execution time for mining sequence patterns from each cluster of events. 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 an 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 months), we would apply more efficient pattern mining algorithms such as FPGrowth [22] and PrefixSpan[23]. 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.

5.4. *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 traffic 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 Section 5.3, 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 gap between user access behaviors and existing access control policies:

- 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. Discussion

The quality of resulting behavior patterns discussed in this paper is controlled by the parameter "*minsup*" in both association mining and sequential pattern mining engines and the useful knowledge about building constrained clusters. As demonstrated, the tool produces high-quality results with a recommended initial seed event and user produced constraints based on suggested significant attribute values. The tool allows the user to stop the process and reproduce cluster constraints if the resulting behavior pattern is unsatisfactory. It also allows user to remove the previously analyzed clusters from dataset, and the tool can provide new recommendations on the basis of the present situation. For this paper, we examined the proposed method with a middle-size system (300+ users); however the empirical results show that the process will also terminate in a reasonable time for large systems through reduced search space, constrained-clusters and user involvement. In addition to knowledge recommendation in the phase of grouping highly related events, we plan to extend our work to provide step-by-step guidance throughout the whole process such as: i) investigating the characteristics of the extracted behavior patterns and committing recommendations to identify common behavior and abnormal behavior; ii) detecting system security policy vulnerabilities and providing reasonable advice on policy refining.

7. Conclusion

In this paper we proposed a knowledge-driven decision support system that effectively assists and guides system administrators to obtain deep insight into the dynamic behavior patterns of a large number of system users. The model is generic in the sense that it allows the user to map domain specific audit data schema onto a standard internal attributed event and behavior model. The recommendation tool supplies pre-processed event and attribute information, using data mining and association-based similarity metrics, to enable the user to acquire some valuable features of the dataset. The proposed association-based event similarity metric measures the maximal association among events as the means to gather highly related events. A behavior pattern query language is designed, allowing the user to compose queries for bringing together interested events and identifying complex behavior patterns. Based on the constrained-clusters, sequential pattern mining is employed to extract frequent behavior patterns in accordance with the proposed behavior model through a context-based action sequence with time interval. Finally, the recommendation system helps system administrators to explore the opportunities to refine the existing security policies by the means of analyzing salient features and characteristics of discovered behavior patterns. Experiments with a middle size distributed medical imaging system provided an evaluation of the proposed approach that proves the knowledge extraction and utilization is crucial in improving the quality of discovered behavior patterns.

Appendix A. Keywords of BPQL

Extended Backus-Naur Form (EBNF) notation is used for denoting the syntax of the proposed Behavior Pattern Query Language.

- 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

Table 5. Keywords of BPQL

Keywords	**Description**
BEGIN-BPQL	BPQL starts.
END-BPQL	BPQL ends.
EVENT	An atomic user-system interaction.
CLUSTER	A group of related events satisfying some constraints.
MAIN-SEED	The initial assigned event into the cluster.
INTRA-CONSTRAINT	The constraints are applied between actions in the same cluster.
INTER-CONSTRAINT	The constraints are applied between actions from different clusters.
GROUP BY	Group the results in terms of the following attributes.
HAVING	Used together with aggregate functions.
SUM	Sum value of specified attribute.
AVG	Average value of specified attribute.
COUNT	The number of actions that are satisfied.
MIN	Minimum value of specified attribute.
MAX	Maximum value of specified attribute.

Appendix B. Syntax of BPQL

<BPQL_query> ::= BEGIN-BPQL <cluster_specification> END-BPQL
<cluster_specification> ::= {<cluster_statement>} {<inter_cluster_constraint>}
<cluster_statement> ::= <cluster_name> <event_main_seed>
<intra_cluster_constraint>
<cluster_name> ::= CLUSTER: <cluster_identifier>
<event_main_seed> ::= MAIN-SEED: EVENT <event_identifier>
<intra_cluster_constraint> ::= INTRA-CONSTRAINT: {<cluster_constraint_expr>}
<inter_cluster_constraint> ::= INTER-CONSTRAINT: <cluster_scope>
{<cluster_constraint_expr>}
<cluster_scope> ::= {CLUSTER <cluster_identifier> ',} CLUSTER <cluster_identifier>

$<cluster\_constraint\_expr>::= \{< attribute\_constraint > | < aggregate\_attribute\_constraint >\}$
$< attribute\_constraint >::=< attribute\_name >< operator >< attribute\_value >$
$< aggregate\_attribute\_constraint >::=< group\_expr >< aggregate\_constraint\_expr >$
$< group\_expr >::= GROUPBY\{< attribute\_name >',' \} < attribute\_name >$
$< aggregate\_constraint\_expr >::= HAVING < aggregate\_function >$
$< operator >< integer >$
$< aggregate\_function >::= SUM|AVG|MIN|MAX|COUNT$
$< attribute\_name >::=' User'|'Role'|'Location'|'Operation'$
$|'Date'|'Resource'|'Time'$
$< operator >::='=' |' \neq' |' \geq' |' \leq' |' >' |' <'$
$< attribute\_value >::=< identifier > | < integer > | < bool >$
$< event\_identifier >::=< identifier >$
$< cluster\_identifier >::=< identifier >$
<identifier> ::= a string of character
<integer> ::= an integer value
<bool> ::= a boolean value

References

[1] Deris Stiawan, Mohd. Yazid Idris, Md. Sah Hj Salam, and Abdul Hanan Abdullah. Intrusion threat detection from insider attack using learning behavior-based. In *International Journal of Physical Sciences 7.4*, pages 624–637, 2012.

[2] Yuval Elovici Chanan Glezer Shabtai Asaf, Uri Kanonov and Yael Weiss. "Andromaly": a behavioral malware detection framework for android devices. In *Journal of Intelligent Information Systems 38, no. 1*, pages 161–190, 2012.

[3] Qing Ye, Xiaoping Wu, and Bo Yan. An Intrusion Detection Approach Based on System Call Sequences and Rules Extraction. In *In e-Business and Information System Security (EBISS), 2nd International Conference on IEEE*, pages 1–4, 2010.

[4] Yi Hung Lin. Anti-fraud system and its method for on-line transaction with credit card. In *U.S. Patent 20,150,032,618*, 2015.

[5] Mingmin Yang Yuan Li Haixin Ma Weining Qian Zhigang Cao Guan Wanqiu, Haoyu Gao and Xiaoguang Yang. Analyzing user behavior of the micro-blogging website Sina Weibo during hot social events. In *Physica A: Statistical Mechanics and its Applications 395*, pages 340–351, 2014.

[6] Domenico Cotroneo Cinque Marcello and Antonio Pecchia. Event logs for the analysis of software failures: A rule-based approach. In *Software Engineering, IEEE Transactions on 39, no. 6*, pages 806–821, 2013.

[7] Mohammad H. Yarmand, Kamran Sartipi, and Douglas G. Down. Behavior-based access control for distributed healthcare systems. In *Journal of Computer Security*, pages 1–39, 2013.

[8] Kamran Sartipi, Krupa A. Kuriakose, and Weina Ma. An Infrastructure for Secure Sharing of Medical Images between PACS and EHR Systems. In *International Conference on Computer Science and Software Engineering*, pages 245–259, 2013.

[9] Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. Frequent pattern mining: current status and future directions. In *Data Mining and Knowledge Discovery*, pages 55–87, 2007.

[10] Rakesh Agrawal, Tomasz Imieliski, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record. Vol. 22. No. 2. ACM*, 1993.

[11] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Data Engineering. IEEE*, 1995.

[12] Minos N. Garofalakis, Rajeev Rastogi, and Kyuseok Shim. SPIRIT: Sequential pattern mining with regular expression constraints. In *VLDB. Vol. 99*, 1999.

[13] Chung-Ching Yu and Yen-Liang Chen. Mining sequential patterns from multidimensional sequence data. In *Knowledge and Data Engineering, IEEE Transactions*, pages 136–140, 2005.

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

[15] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. In *Data Mining and Knowledge Discovery*, pages 259–289, 1997.

[16] Power Daniel J. Decision support systems: concepts and resources for managers. In *Greenwood Publishing Group*, 2002.

[17] Murtadha M. Hamad and Banaz Anwer Qader. Knowledge-Driven Decision Support System based on Knowledge Warehouse and Data Mining for Market Management. In *Global Journal of Management And Business Research 13.10*, 2014.

[18] MARC-HI Everest Framework. http://te.marc-hi.ca/view.aspx?project=af66d54ed41e4ac18b44d0d3ca6cabf0.

[19] RFC 3881 - Security Audit and Access Accountability Message XML Data Definitions for Healthcare Applications. http://tools.ietf.org/html/rfc3881.

[20] Liu Bing, Wynne Hsu, and Yiming Ma. Mining association rules with multiple minimum supports. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining.ACM*, 1999.

[21] Gephi - The Open Graph Viz Platform. http://gephi.github.io/.

[22] Jian Pei Han Jiawei and Yiwen Yin. Mining frequent patterns without candidate generation. In *In ACM SIGMOD Record, vol. 29, no. 2*, pages 1–12, 2000.

[23] Behzad Mortazavi-Asl Helen Pinto Qiming Chen Umeshwar Dayal Pei Jian, Jiawei Han and Mei-Chun 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.