A CRIME INVESTIGATION TOOL BASED ON EVENT PATTERN DETECTION :

by

Osbert Osamai

Reg. No: 2010/HD18/1259U

BITC (KYU), DCS (KYU)

Department of Networks

School of Computing and Informatics Technology, Makerere University

E-mail: omeja4ever@gmail.com,oosbert@trobank.com

Tel: +256712738530

A Project Report Submitted to the School of Graduate Studies in Partial Fulfillment for the Award of Master of Science in Data Communication and Software Engineering Degree of Makerere University.

OPTION: Mobile Computing

Supervisor

Kanagwa Benjamin (PhD)

Signature ———————————

Date ———————————

Department of Networks

School of Computing and Information Technology, Makerere University

Email: bkanagwa@gmail.com Tel: +256-712495020

March 2013

Contents

LIST OF ACRONYMS

CEP	Complex Event Processing
SNA	Social Network Analysis
GIS	Geographical Information Systems
SPP	Spatial Point Patterns
SQL	Structured Query Language
SOA	Service Oriented Architecture
JDBC	Java Database Connectivity
ODBC	Open Database Connectivity
SOA	Service Oriented Architecture
XML	Manipulation Language
CSV	Comma Separated Values
POJO	Plain Old Java Object
SDLC	System Development Life Cycle

1 Introduction

1.1 Background

Crime investigation world over is considered a difficult and laborious process that heavily relies on efficient crime analysis to ensure accurate and timely conclusions. This is not helped by the fact that law enforcement agencies deploy manual investigation processes and computerized systems that cannot quickly identify complex crime patterns.

In Uganda, the situation is no different with these agencies facing massive delays in crime solving and increasing numbers of sophisticated crimes, most of them of the organized. Every tens of thousands of cases are carried forward to the following year, uncompleted. As the usual circle of crime would dictate, fresh cases are reported every day, and, gradually, older cases left uncompleted lose the urgency they initially generated and, inadvertently, they die a natural death [21].

To avert this problem there is need to deploy sophisticated tools, technologies and resources that can enable crime investigators quickly reach reasonable conclusions by identifying patterns of behavior in criminals. In so doing, not only will crimes be investigated faster but some future crimes may be prevented based on recurring patterns identified. However, intelligence and law enforcement agencies are often faced with the dilemma of having too much data, which in effect makes too little value. On one hand, they have large volumes of raw data collected from multiple sources: phone records, bank accounts and transactions, vehicle sales and registration records, and surveillance reports. On the other hand, they lack sophisticated network analysis tools and techniques to utilize the data effectively and efficiently [25].

In this project the phenomenon of Complex Event Processing (CEP) is used to detect patterns in crime related events using the Esper engine for high throughput and performance. CEP analyses low level events to produce a single complex event and has been successfully used in Stock Trading, Network Analysis and other areas.

1.2 Statement of the problem

Due to the manual nature of crime investigations in Uganda, case backlog remains a critical issue leading to delayed justice, unpunished criminals and of course tainting the image of the agencies. These manual methods often lead to fatigue, poor statistical analysis and the inability to solve crimes through pattern detection and analysis.

Criminals have become very intelligent due to the advancement of technology and therefore they conduct crimes in an untraceable manner. Majority of those crimes evolve in a long period of time making them even more difficult to predict. Therefore the rate of organized crime is on the rise most of which are orchestrated in vast geographical areas using these complex techniques.

Therefore, manual techniques of analyzing such data with a vast variation have resulted in lower productivity and ineffective utilization of manpower [12].There is need to develop a tool to quicken the investigation process by accurately guiding investigators in evidence analysis and also use recurrent patterns to prevent future crimes.

1.3 Objectives

1.3.1 General Objective

The objective of this project is to design, develop and deploy a crime investigation tool to guide investigators by detecting crime patterns and monitoring criminal activities. The tool will operate by filtering criminal activities as events and detecting predetermined patterns. This will not only reduce on time spent during investigations but also prevent some crimes from occurring by discovering recurring patterns.

1.3.2 Specific Objectives

Specific objectives include:-

1. Design and build a crime investigation tool based on complex events with a processing engine and web-based visual interface.

2. Thoroughly validate and test the solution.

3. Deploy the solution in a live working environment.

1.4 Scope

The project covers general aspects of using complex event processing in applications but the implemented tool processes data from an existing information system and only focuses on crime patterns inherent in financial embezzlement related crimes. Document processing and forensics related actions are not covered in this project.

1.5 Significance of Project

The benefits of the project apply to the Police and the general public. These include:-

1. Improve efficiency by reducing manual operations and concentrating on analysis and prediction efforts hence also reducing on the time spent during investigations.

2. Reduce on crime rates by preventing their occurrence through discovery of uniform patterns.

2 Literature Review

This section provides a general literature review of major data mining techniques used in existing crime investigation systems and Complex Event Processing as the preferred technology for this project.

2.1 Current Technologies in Crime Investigation Systems

Existing crime investigation systems tend to vary in terms of their overall capabilities and technical operation. In one study [2]the existence of prominent criminal investigation software like HOLMES2, BRAINS and Analysts' Notebook which are used by criminal analysts in the United Kingdom and Holland was acknowledged. This category can be classified as early generation systems that mainly focused on analyzing evidence separately without linking multiple sets of evidence in order to solve crimes. They were also not designed to communicate with existing case management systems to facilitate data exchanges.

The Second generation systems around the world relied heavily on data mining techniques to query large datasets for meaningful patterns in order to help investigators solve crimes. These methods however fall short in terms of supporting decision making largely due to poor processing speeds and querying algorithms.

Link Analysis is a technique used in data mining. These tools have for long been used by law enforcement agencies to identify, analyze and visualize relationships between crime entities. In a study [17] , it is revealed that through association paths linking suspects and victims in crime, link analysis discovers information about motives and hence provides investigative leads.

Another technique used in crime pattern detection involves several data mining steps like hotspot detection, crime clock, crime comparison and crime pattern visualization. Numerous algorithms are used to relate multiple crime scenes, represent a number of crimes scenes on a daily basis, compare different crimes to estimate growth rates and visualize the changes in crime occurrence frequencies [12].

A study on crime network analysis [25] suggests that law enforcement agencies need to deploy reliable data and sophisticated tools as critical tools in the discovering useful patterns in data. They introduce a data mining techniques called Social Network Analysis (SNA) which is used to discover hidden patterns in large volumes of crime related data. An approach of SNA referred to as black modeling is used in criminal networks to reveal associations between subgroups based on a link density measure. Discovery of new structural patterns during this process can enable prevention of crimes and also modify conventional view of certain crimes by investigators.

2.2 Related work in Uganda

In Uganda, some studies [20] have been conducted in the area of crime investigation. one research discusses a model for forensic investigations that performs detection of incidents through system monitoring and performs data analysis to unearth the crime scene, suspect and how the crime was perpetrated. The study proposed a new model based on five iterative phases that were meant to strengthen the crime detection and analysis process.

Another study [15] on crime prevention suggested a combined application of data mining techniques alongside GIS (Geographical Information Systems) to discover crime data in disorganized settings like Uganda. Spatial point patterns (SPP) based on coordinates of events such as locations of crime incidences and the time of occurrence are used.

2.3 Complex Event Processing

A Complex event is an event that abstracts or aggregates simple (or member) events [10]. Simple and complex events are normally represented in linear ordered sequences called Event Streams. These streams are usually bound by time intervals and may contain different types of events.

Complex Event Processing (CEP) is defined as the process of detecting complex events using continuously incoming events on a lower abstraction level [3]. This study justifies the need for CEP given the fact that single events on their own may not be sufficient in determining certain patterns. CEP therefore provides a platform for a combination of events to be processed and

analyzed.

CEP is a foundational technology for detecting and managing the events that happen in event driven enterprises. It is a collection of methods, tools and techniques applied in processing events as they happen. In order to achieve a lot from CEP, happenings of events in enterprises need to be well understood. This can be achieved by organizing events into structures or hierarchies, identifying relationships among events (causal, time, aggregation) and organizing events in different views from different personnel. In CEP, higher- level knowledge is derived from lower-level events which are a combination of various occurrences. CEP can be viewed in two types, the first one involving specification of complex events as patterns and detecting them effectively, whereas the other type involves detecting new patterns as complex events. In the first case, event query languages offer convenient means to specify complex events and detect them efficiently. In the second case, machine learning and data mining methods are applied to event streams.

Detection of complex events is, of course, no an end in itself; an event-driven information system should react automatically and adequately to detected events. Typical reactions include notifications (e.g., to another system or a human user), simple actions (e.g., buy stocks, activate fire extinguishing installation), or interaction with business processes (e.g., initiation of a new process, cancellation or modification of a running process).

A study about the history of CEP [9] traces its roots to university and company research groups in the late 90's which were involved in the areas of active databases, event driven simulation, networking and event processing in middleware. This explains partly why the CEP query languages are based on SQL syntax having been influenced by research on active databases. CEP products at that point did not generate much interest until the late 2000's when CEP was deployed as add-ons on SOA architectures and ESBs.

Databases are distinct from event queries used in CEP because of the latter's ability to continuously detect events as they happen rather than just acting on stored datasets. Event processing languages need to enable the possibility of joining several individual events together, so that their combined occurrences over time yield a complex event and complex events must contain the element of time, to track times when events occur.one study introduced the need for revision of events in cases of erroneous data. In practice, there are a number of reasons requiring revisions in event stream

processing. For example, an event was reported by mistake, but did not happen in reality (and the mistake was realized later); an event happened, but it was not reported (due to failure of either a sensor, or failure of the event transmission system); or an event was triggered and later revoked due to the transaction failure. Also very often streaming data sources contend with noise (e.g., financial data feeds, Web streaming data, updates etc.) resulting in erroneous inputs and, therefore, erroneous complex event results.

Through Complex Event Processing (CEP), companies and organizations can manage processes in close to real-time. It is however noted that due to the complexity of generic event processing frameworks offered by the industry, the configuration and setup of CEP applications are left to external experts who are more knowledgeable in complex event logic. A CEP application retrieves events for all noteworthy incidents in the business environment. In various parts of the application, event-pattern rules are applied on the incoming event stream to detect relevant patterns, e.g., an uptrend in application errors or execution delays. In response to such patterns, the CEP engine proactively intervenes in the business environment, e.g., by temporarily allocating additional resources, throttling uncritical business tasks or notifying system administrators [9].

Pattern matching is a key feature of all CEP technologies which involves finding subsets of data matching a given pattern and also relationships between those subsets. A study about CEP under uncertainty explains the role pattern matching plays in allowing users to look beyond individual events and find specific collection sets.

Solution templates are proposed to perform data mining procedures on historical data to identify event patterns besides real-time monitoring. The central concept of any template's event processing infrastructure is the event processing map, a predefined orchestration of event adapters and event services. Event adapters may be considered the actual interface to the underlying source system: Depending on their implementation, event adapters translate real-world actions (such as a user actually placing a bet in an online gambling platform) into event representations of a certain event type, and vice versa. Event services receive events from event adapters or other event services, process them based on implementation of specific logic, and respond back to the map.Detecting events is based on some considerations like, some events sharing time elements, the order of events, time bounds within events and detection of events of long time lags. To gain insight into processes there is need to include the following components in CEP application, Facilities (graphical or

textual) for precise description of complex patterns of events, Scalable performance, modular rules engines to detect complex patterns of events, Facilities for defining and composing event pattern triggered rules for pattern abstraction.

2.4 Event Processing Languages

An analysis of Event Processing Languages [3] revealed the need to shift from using general-purpose languages like C, Java, C++ e.t.c for CEP applications due to low-level complexities. Using such languages along with complexities like data structures and algorithms can only complicate the development process. The study provided a detailed analysis of existing CEP programming Languages and platforms based on their expressivity and integration capabilities. Expressivity is measured by one of the following abilities, filtering streams by event type, processing a subset of events (windows), data extraction and aggregation of data over events, performing conjunctions and disjunctions, show temporal relations between events, showing causality of events, negation and counting of events, event instance selection and consumption to prevent reuse of events in pattern detection and integration of event data and non-event data (data from outside). Languages are also grouped into the categories of data stream query languages, composition-operator-based languages, production rule languages and logical formulas.

STREAM (Stanford Stream data Manager) is a language whose focus was to develop methods to manage and query data in data streams and was a result of a research project at Stanford University. The project also produced a CEP engine called STREAM and an Event Query Language called Continuous Query Language to query events. STREAM was a basis for other data stream languages like Esper and its querying syntax resembles SQL very strongly.

Borealis is a CEP engine developed at Brandeis University and MIT that uses a ”boxes and arrows” approach. Queries are described graphically with queries as boxes and streams as arrows connecting boxes. The approach was first used in an earlier engine, Aurora. The main difference between Borealis and stream languages is the focus on query evaluation that Borealis offers resulting in less abstract queries than STREAM.

Active Middleware Technology (AMiT) enables IBM middleware to become event-based. This

technology is implemented in several products, most notably extending Web- Sphere Broker with CEP capabilities. As WebSphere is a commercial product, it is not freely available (requires registration .Basic events are declared with their attributes in event tags. Lifespans are windows defined by two events, an initiator and a terminator event. Lifespan types are therefore declared by referencing start and end event types. Whenever an event matching the initiator speci?cation is detected, a new lifespan of this type is opened, and when an event matching the terminator specification is detected, the lifespan is closed. Complex events are called situations. A situation consists of at least one data attribute (it has to carry at least one kind of information), exactly one operator, and a lifespan type. Situations are only tried to be detected in lifespans of its type. A lifespan may be referenced by multiple situations

RuleCore is a CEP engine developed by Analog Software, building on research at the University of Skyde. As the name suggests, rules are the central concept of ruleCore. The ruleCore engine processes events using ECA (Event-Condition-Action) rules that consist of three parts: for every event (basic or complex), check a condition; if it is true, execute the action. ruleCore has two implementations; an open source variant called ruleCore, released under the terms of the GPL; as well as a commercial version called ruleCore CEP Server. RuleCore uses so-called detector trees for event detection. Leaf detector nodes (detector nodes without children) detect single events (they pick up events of their type). They are inactive until an event of their type is delivered to the rule (usually by entering the system, although exceptions are mentioned in the next paragraph), after which point they are always active. To detect complex events, a detector tree is built: the leaves detect simple events, and inner nodes detect complex events depending on whether its children detected events.

SASE+ is a CEP system developed at the University of Massachusetts, Amherst. It is an extension of the older SASE system. The system is designed for event streams with many events per time unit and also queries using large time windows, creating new issues regarding efficient query execution. The project's purpose is to devise techniques for high-performance querying of event streams, using a declarative, composition-operator-based language. Although SASE+ is an agile language and concentrates only on pattern matching on streaming data, the pattern matching properties of SASE+ can be used in more general contexts.

Esper is an open-source CEP engine, developed by EsperTech Inc. and volunteers, released under

the GNU General Public License (GPL v2). As stated on the official web site, it is designed for CEP and Event Stream Processing (ESP).There are two implementations of Esper, Esper for Java and NEsper for .NET. Both supply an API to access the engine features, such as deploying queries, sending events into the engine and retrieving events out of the engine, in their respective language. Events are objects in their respective language; for Esper, events can be instances of java.util.Map,org.w3c.dom.Node (Java representations of XML documents), or other Java objects. Regardless of the implementation language, queries are stated in a SQL-like language called Event Processing Language (EPL).

Cayuga is a research CEP engine developed at Cornell University. It sets itself apart from other engines in that it deliberately sacrifices expressivity for performance, targeting applications running large numbers of queries. It is free software, available under the terms of the BSD license. Cayuga uses an Event Query Language called Cayuga Event Language (CEL). While its syntax resembles SQL, like many data stream languages, it also offers patterns, although using a different approach compared to Esper, inspired by regular expressions.

Drools, also known as JBoss rules, is a production-quality business rule management system, including a production rule engine. It is free software, released under the Apache License. The Drools engine is implemented in Java, as is JBoss, and is also controlled using that language. Initialization of the engine and deployment of rules is implemented in Java. Also, as unusual for production rule engines, rules never fire by themselves, but are issued to do so by the Java program that controls the engine. In addition, Drools can be extended by defining so-called Domain Specific Languages. These are languages that may have a different syntax than the standard Drools syntax to write queries in. Rules in Domain Specific Languages are then translated into the Drools language when inserted into the engine.

XChangeEQ is a research Event Query Language. It is developed at the University of Munich and designed for automated reasoning on the Semantic Web. XChangeEQ introduces a new style of event querying. It separates event query features into four so-called dimensions: Data extraction, event composition, temporal relationships, and event accumulation. Most operators belong to exactly one of these dimensions. This was done to define clear semantics. As XChangeEQ is designed for use on the Web, it works best at processing tree-structured events, such as XML messages. Queries are generally structured like the XML representations of the events queried.

For querying simple events, it embeds the Xcerpt language. Xcerpt queries apply patterns to XML documents, similar to templates. Change is a reactive programming language. Using Event-Condition-Action rules, it allows Web sites to react to changes at other Web sites, for example by updating its own data.

TelegraphCQ, from the University of California at Berkeley, is designed to provide event processing capabilities alongside relational database management capabilities by utilizing the PostgreSQL open source code base. The existing architecture of PostgreSQL is modified to allow for continuous queries over streaming data. Several components of the PostgreSQL engine underwent very little modification, while others were significantly changed. The most significant component of the TelegraphCQ system is the ”wrapper,” which allows for data to be pushed or pulled into the Telegraph processing engine, and custom wrappers allow for data to be obtained from any data source .

BEA Systems in 2007 introduced of their WebLogic Real Time and WebLogic Event Server systems. More specifically, their Event Server technology is a focus on event-driven service oriented architecture which provides a response to events in real-time. As part of the package, they provide a complete event processing and event-driven service-oriented architecture infrastructure that supports high-volume, real-time, complex, event-driven applications. This is one of the few commercial offerings of a complete, integrated solution for event processing and service-oriented architectures.

Truviso is a commercial event processing engine that is based on the research toward the Telegraph CQ project at UC Berkeley. The”claim to fame” for Truviso is that it supports a fully functional SQL, and integrates PostgreSQL relational database alongside a stream processing engine. The integration of PostgreSQL leads to other aspects of the Truviso system. The queries are simply standard SQL with extensions that add functionality for time windows and event processing. Carried over from PostgreSQL are user-defined functions, as well as JDBC and ODBC interfaces. In addition, the use of an integrated relational database allows for easy caching, persistence, and archival of data streams, as well as queries that include not only real-time data, but also the historical data [7].

3 Methodology

The development of the system followed the SDLC development methodology.In addition, object oriented approach was used in design and implementation.

3.1 Requirement Gathering

3.1.1 Interviews

Interviews were conducted with investigative officers at Police headquarters, Jinja Road police station and the Special Investigation Unit, Kireka. This was carried out to ascertain the nature of financial crime cases, the investigation processes involved and the criteria used to zero down on a suspect. Telephone interviews were also conducted in situations where physical access was challenging.

The questions asked were related to the following:-

1. Type of evidence gathered and used in such investigations.
2. The patterns which are common in most cases and which the system will base upon.
3. Any existing systems used for crime investigation and formats of existing crime records.

3.1.2 Existing Literature

Existing literature on Complex Event Processing was read to determine the best design approach for the project and to discover the benefits of using the CEP model.Implementations of the Esper engine in particular were studied to provide a benchmark for the project implementation.

The Annual Police report was also read to acquire an in depth understanding of the impact of crimes to society and government. Case statements some handwritten were also analyzed and these provided guidance on data to capture and track in the system.

3.1.3 Questionnaires

Questionnaires were distributed to investigation officers and their superiors to determine the need for such a tool and the benefits it would bring to the Police force. The returned forms portrayed a clear need for the system as it would reduce on processing time and reduce case backlog.

3.2 Design Methodology

This section describes the system environment, interactions, requirements, architecture and input/output formats as well as processing logic. The system was modeled using UML (Unified Modeling Language) tools based on the object oriented design approach.

3.2.1 System Overview/Context Diagram

At the highest level, the system interfaces with an existing crime database system in order to retrieve records which are converted to events and processed.

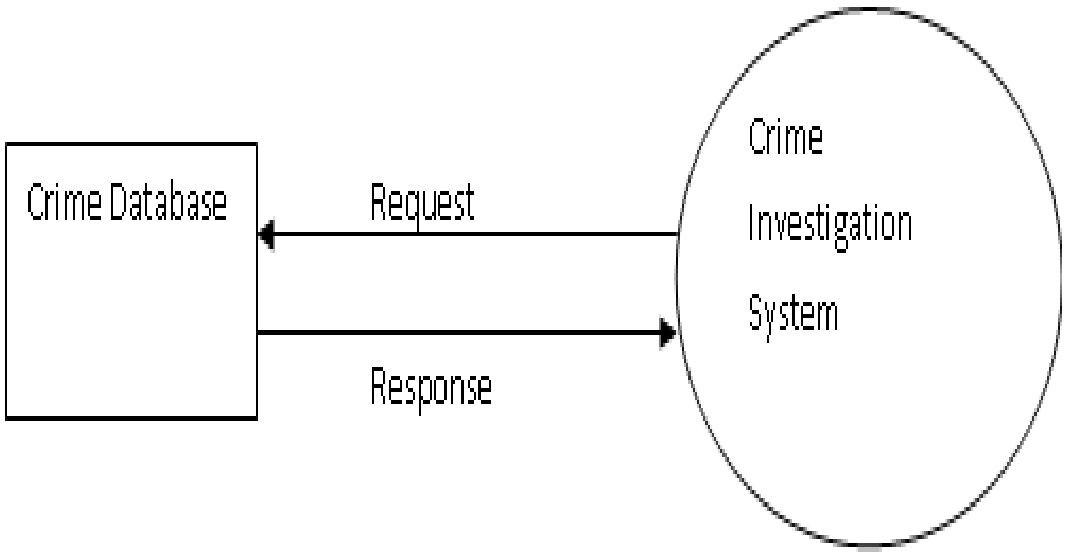

Figure 1: Data flow Diagram

3.2.2 System Architecture

The architectural design comprises of the external database which provides input data, the application server which describes business logic and links all components (Client, Database and Web Server), the web client and the web server that processes requests and provides responses.

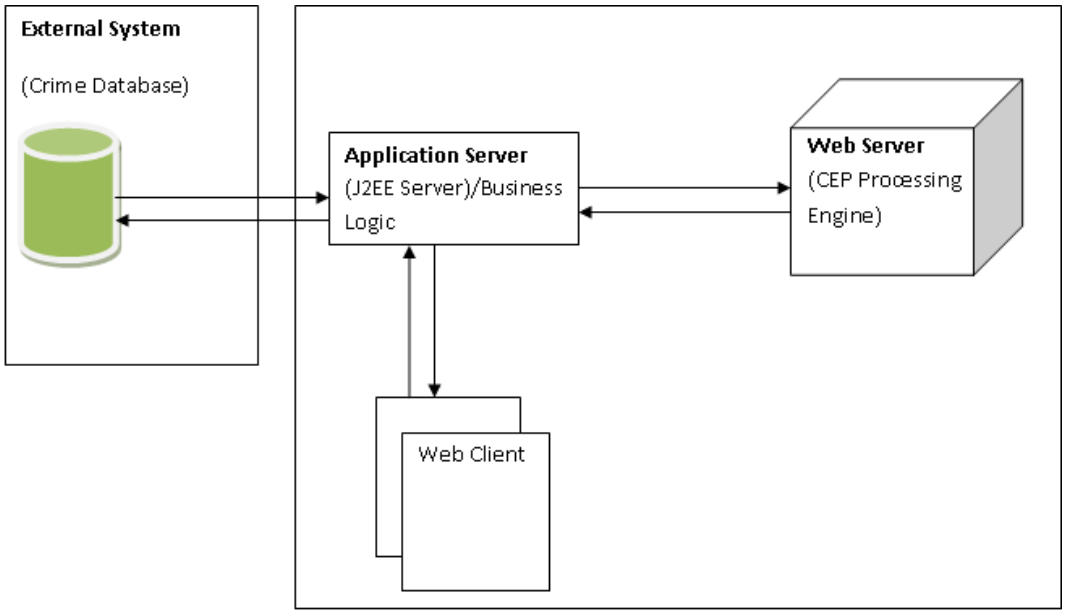

Figure 2: Data flow Diagram

3.2.3 Data Flow Diagram

The process was designed to start from the web client (user) making a request for records from and existing database. The returned data is sent to the processing engine and checked against defined pattern. Based on the results a message is displayed to the client along with records (in the case of a successful pattern match).

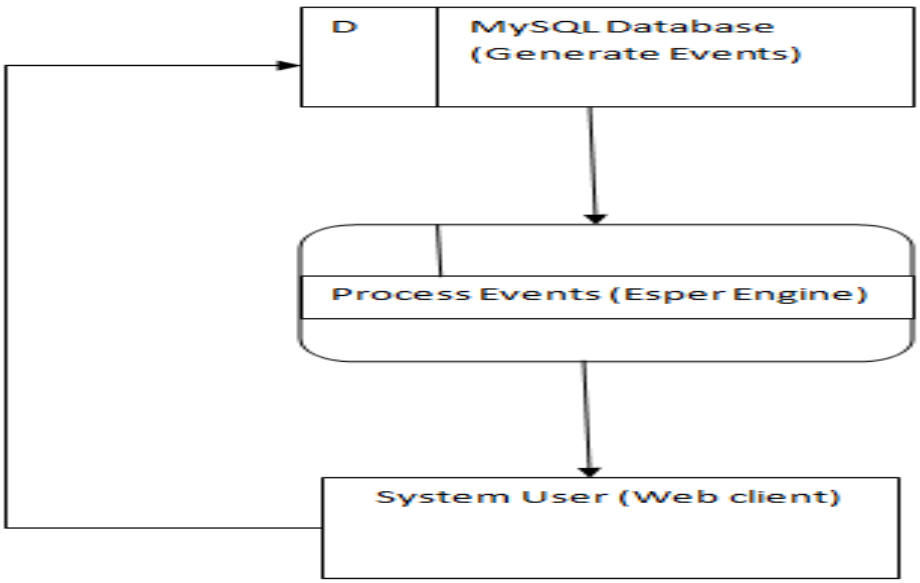

Figure 3: Data flow Diagram

3.3 Implementation Methodology

The tool was implemented as a web-based application to enable access by users throughout the Police network and to provide capabilities for access by mobile devices as well. Other reasons for choosing a web application include; OS platform independence, ease of maintenance and limited upgrades from the client side. It was built on the Java EE platform using the glassfish application server and the JSF 2.1 framework.

3.3.1 Java EE Platform

The Java EE platform uses a simplified programming model. XML deployment descriptors are optional. Instead, a developer can simply enter the information as an annotation directly into a Java source file, and the Java EE server will configure the component at deployment and runtime. These annotations are generally used to embed in a program data that would otherwise be furnished in a deployment descriptor. With annotations, you put the specification information in your code next to the program element affected. In the Java EE platform, dependency injection can be applied to all resources a component needs, effectively hiding the creation and lookup of resources

from application code. Dependency injection can be used in EJB containers, web containers, and application clients. Dependency injection allows the Java EE container to automatically insert references to other required components or resources, using annotations [5].

3.3.2 JSF Framework

Java Server Faces technology is a user interface framework for building web applications. The main components of Java Server Faces technology are as follows: A GUI component framework, a flexible model for rendering components in different kinds of HTML or different markup languages and technologies. A Renderer object generates the markup to render the component and converts the data stored in a model object to types that can be represented in a view, a standard Render Kit for generating HTML/4.01 markup. The features supporting the GUI components include; Input validation, Event handling, Data conversion between model objects and components, Managed model object creation, Page navigation configuration and Expression Language (EL). For this project the Java EE 6 platform was used with Java Server Faces 2.1 and Expression Language 2.2 [5].

3.3.3 The Client Tier

These are web pages developed using XHTML and using the Expression Languages to connect the business layer (Managed Beans). The prime faces library was used to provide a desktop look and feel to the interface. Primefaces is a lightweight library that hides complexities from user while providing extended capabilities to the default XHTML controls as well as quick software development.

3.3.4 The Java EE Server

This tier contains business logic in form of managed beans (POJOs) interacting with the MySQL database through the Java Persistence API (JPA) 2.0 and servlets.The Java Persistence API (JPA) is a Java standards-based solution for persistence. Persistence uses an object/relational mapping

approach to bridge the gap between an object-oriented model and a relational database. The Java Persistence API can also be used in Java SE applications, outside of the Java EE environment. Java Persistence consists of the following areas; the Java Persistence API, the query language and Object/relational mapping metadata [23].

3.3.5 The EIS (Extended Information Server) Tier

This is the external system that provides crime records to be analyzed. It is JSF web application running a MySQL Server 5.5 database. The relevant queries were written to return records to the main system through the JPA entities.JPA entity classes are generated from the database tables and these then provide a link to the crime investigation web application via the JPA.

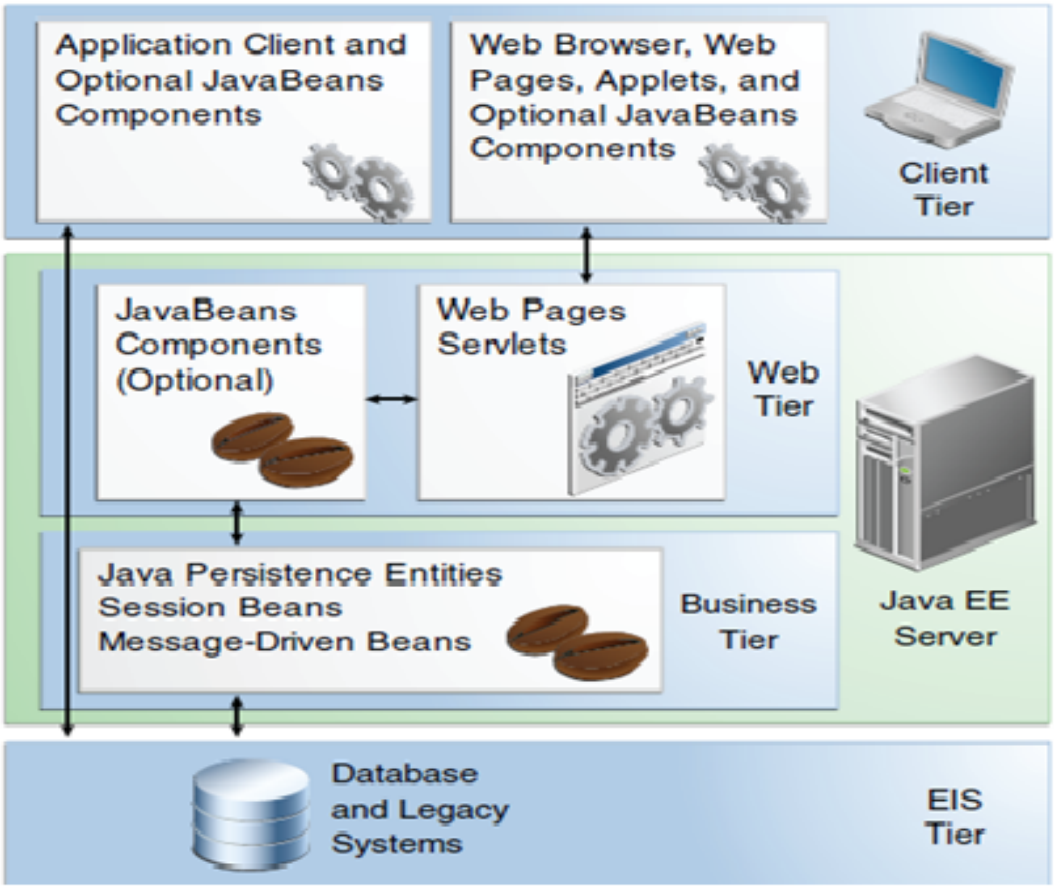

Figure 4: Data flow Diagram

3.4 Testing

Junit was used to test the system for the state of managed beans, confirming if the navigation commands lead to the correct pages, checking if invalid error messages appear, testing application configurations and database connections.

4 The Crime Investigation Tool

The system consists of four components at the highest level. These are the web client, processing engine (Esper), data source (MySQL Database) and application server (glassfish).This chapter explains how these components interact at a the highest level to process records and display to the user.

4.1 System Processes

A web user interface is used by crime investigators to select records of particular cases to be sent to the engine for processing and also to receive processed responses on web pages. The client communicates with the database to generate input data and interacts with managed beans (business logic) to send and receive events from the processing engine.

1. A user requests for records related to a particular case through the web interface. The records from the data source are sent to a processing engine that filters incoming events based on defined crime patterns.

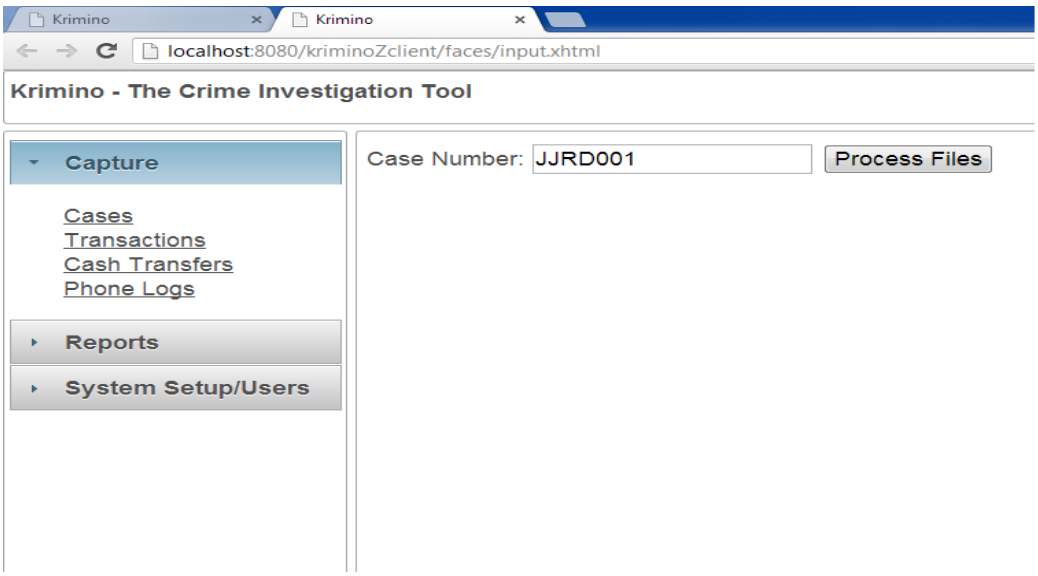

Figure 5: Data flow Diagram

2. The engine processes the records generated from the database and sends regular responses to the web client showing events that have matched the described pattern. The engine implements a listener which constantly receives events as they stream in.The crime pattern is determined by the investigator and can be adjusted through the user interface.Records requested by the client are generated in CSV (Comma Separated Values) format and sent to the engine for processing. The records are stored in a MySQL Database for all cases that are being investigated. The events are represented in POJO (Plain Old Java Object) format with getters and setters in a form understandable to the CEP engine.

3. Based on the processed data, the results are displayed to the web page and can be exported in various formats or printed.

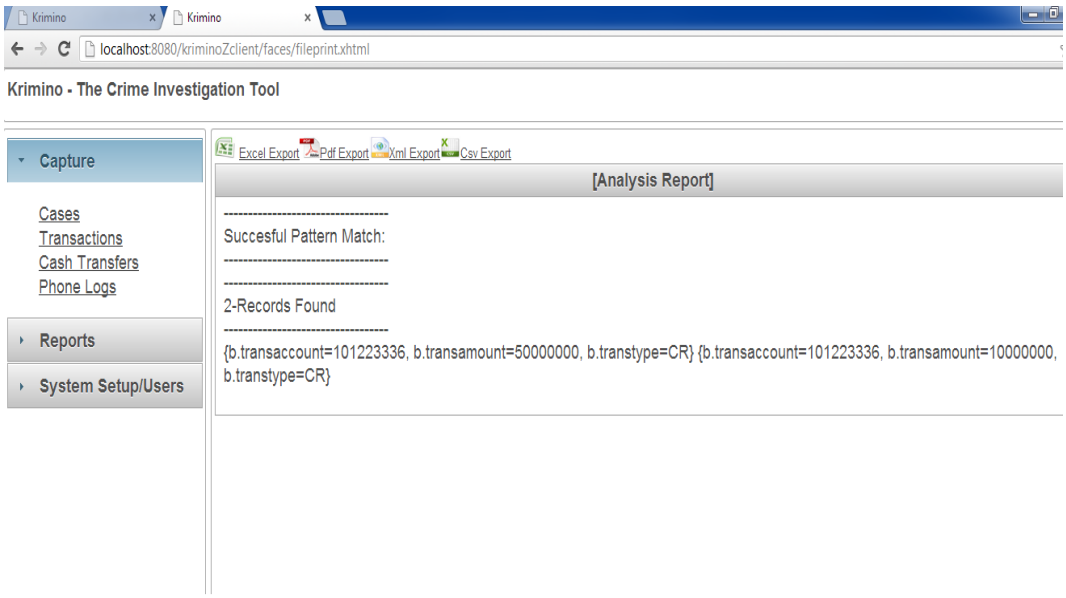

Figure 6: Data flow Diagram

4.2 Patterns Related to Bank Account Activity

Most criminals involved in financially related crimes tend to possess large sums of money in their bank accounts that exceed their source of income. These amounts are deposited on a regular basis (daily, weekly, monthly)in different bank branches and are also used to acquire property within a short period of time.In investigating such crimes, investigators analyze company data detailing the earnings of an individual and the payment dates of his/her salary.They also probe assets acquired by the individual from the past todate detailing accquisition periods and amounts.

Other details investigated include:-

1. Bank Statements for the both the individual and company for selected periods.
2. Company budgets and financial statements.
3. External sources of income of the individual.
4. Financial authorization powers and signatory levels.
5. Funds transfered from accounts within and out of the country.

Theerefore the processing engine was configured to store patterns checking stremed events (bank statements, funds transfers,expenditures) against salary, known assets and external sources of income.

References

[1] D. Anicic, S. Rudolph, P. Fodor, and N. Stojanovic. Retractable complex event processing and stream reasoning. In *Proceedings of the 5th international conference on Rule-based reasoning, programming, and applications*, RuleML'2011, pages 122–137, Berlin, Heidelberg, 2011. Springer-Verlag.

[2] F. Bex, H. Prakken, B. Verheij, and G. Vreeswijk. Sense-making software for crime investigation: how to combine stories and arguments, 2007.

[3] H.-L. Bui. Survey and comparison of event query languages using practical examples, 2009.

[4] Codehaus. Esper reference tutorial, 2013.

[5] O. coporation. The java ee 6tutorial, 2011.

[6] P. Dekkers. Complex event processing-sl-securenet: Intelligent policing using data mining techniques, 2007.

[7] N. Dindar, B. Güç, P. Lau, A. Ozal, M. Soner, and N. Tatbul. Dejavu: declarative pattern matching over live and archived streams of events. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, SIGMOD '09, pages 1023–1026, New York, NY, USA, 2009. ACM.

[8] A. A. Kulkarni. Arcade - abstraction and realization of complex event scenarios using dynamic rule creation. In *Proceedings of the 5th ACM international conference on Distributed event-based system*, DEBS '11, pages 23–28, New York, NY, USA, 2011. ACM.

[9] D. Luckham. A short history of complex event processing1 part 3: the formative years.

[10] D. Luckham. Complex event processing in financial services, 2010.

[11] D. Luckham. complex or simple event processing, 2010.

[12] S. M.A.P. Chamikara, Y.P.R. D. Yapa and J. Gunathilake. Sl-securenet: Intelligent policing using data mining techniques. In *International Journal of Soft Computing and Engineering (IJSCE)*, 2231-2307, Volume-2, Issue-1, 2012.

[13] S. Myles. Criminal investigation case management., 2000.

[14] H. Obweger, J. Schiefer, M. Suntinger, F. Breier, and R. Thullner. Complex event processing off the shelf–.

[15] D. O. P. Okwangale Fredrick R. Survey of data mining methods for crime analysis and visualization, 2010.

[16] PrimeTek. Prime faces users guide 3.5, 2013.

[17] J. Schroeder, J. J. Xu, H. Chen, and M. Chau. Automated criminal link analysis based on domain knowledge. *JASIST*, 58(6):842–855, 2007.

[18] N. P. Schultz-Møller, M. Migliavacca, and P. Pietzuch. Distributed complex event processing with query rewriting. In *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, DEBS '09, pages 4:1–4:12, New York, NY, USA, 2009. ACM.

[19] F. Technologies. Accelerating complex event processing with memory- centric database (mcdb), 2008.

[20] F. Tushabe. Computer forensics for cyberspace crimes.

[21] U.Police. Annual crime and road safety report 2010., 2010.

[22] K. H. Vellani. Crime analysis for problem solving security professionals in 25 small step.

[23] S. Wasserkrug, A. Gal, O. Etzion, and Y. Turchin. Complex event processing over uncertain data. In *Proceedings of the second international conference on Distributed event-based systems*, DEBS '08, pages 253–264, New York, NY, USA, 2008. ACM.

[24] C. Westphal. Anatomy of a financial crime.

[25] J. Xu and H. Chen. Criminal network analysis and visualization. *Commun. ACM*, 48(6):100–107, June 2005.