Practical Benefits of Using Complex Event Processing in a Service Oriented Architecture

T. S. L. Fernando[1], W. W. M. E. P. Gunetileke[1], H. Halgaswatta[1], S. Weerawarana[2], P. Fremantle[3] and I. Perera[1]
[1]Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka.
[2]WSO2 Inc.,Sri Lanka.
[3]WSO2 Inc., United Kingdom.

Abstract— **The merging of the two great technologies of Service Oriented Architecture and Complex Event Processing is a concept that is quickly gaining popularity around the world with mainstream companies such as IBM joining in the trend. This will open a whole new world of opportunities in IT networks and in many SOA related realms. Project SCI-Flex is an effort which has become a part of this concept with the development of a flexible framework supporting SOA and CEP mediation and much more. This paper explores the opportunities brought out by the merging of SOA and CEP and highlights the approach and success we have achieved in our endeavor, as well as looking ahead at the future real world potential of this concept.**

Index Terms— **CEP, EDA, SOA**

I. Introduction

The businesses of today's world are increasingly adapting distributed computing systems for their IT infrastructure.

With the need for expanding your business and your market beyond your physical locale, this has become a necessity. In this environment, businesses are increasingly also looking towards Service Oriented Architecture (SOA) as a solution. The ability to implement complex business functions in a distributed environment at a lesser cost, with an easy setup as well as high flexibility in updating or customizing its structure, interoperability among different components and systems, and many other similar advantages, SOA is quickly gaining popularity among businesses in this field.

From another point of view, with increasingly larger and more dynamic amounts of raw data in businesses to analyze and process, the concept of Complex Event Processing (CEP) is also gaining popularity. CEP basically allows us to bring together large amounts of data and apply customized rules or criteria to identify patterns in it which would normally be impossible for a regular data processing application. Both these concepts therefore are becoming vita for businesses worldwide, and therefore interest in building up greatly towards some sort of merging of the two. CEP at the moment runs on individual applications or locations. There is no way to simply configure it to utilize its processing power over a distributed network environment governed by an SOA. This is the problem we have taken on to tackle – enabling these two great technologies to coexist.

What we have achieved in this regard in our project, SCI-Flex, (which stands for Flexible Integration of SOA and CEP) is building an SOA infrastructure around a CEP system. This has been done in the manner of components such that users may utilize the framework we have built to use a CEP system within a SOA environment. We have taken an SOA enabler; and Enterprise Service Bus or ESB, and built a system into it which will enable it to seamlessly work with a CEP system, thus granting a user the best of both worlds.

Businesses therefore will be able to apply the power of CEP to its own SOA infrastructure, in a solution customized to their reality. There is obviously a large market of opportunities out there for this merging, and we will look at some of them in detail in this paper as well.

Section Two will give more background information on topics used in this research paper to get the reader familiarized with the concepts.

Section Three of this paper will take a look at the combining of CEP and SOA and will explore its benefits and implications.

Section Four will introduce the approach of SCI-Flex in this realm and how it aims to achieve this integration and mediation.

Section Five will aim to add a practical note to this paper by exploring some of the real world possibilities brought on by the merging of these two concepts.

Section Six will then apply the concepts put forth by SCI-Flex and attempt to demonstrate its practical application in a relevant scenario.

Section Seven: Future Work will highlight possible improvements and new developments that could take place in our venture.

Section Eight: Related Work will be a similar section where we will explore systems and implementations that are similar to the objectives we put forth here.

II. BACKGROUND

This section is aimed at familiarizing the reader with some of the major technologies and concepts that we will be talking about in this research paper. Readers who are familiar with the concepts of SOA, CEP and ESBs may skip this section. A brief guide to the rest of this paper is also included.

A. Service Oriented Architecture (SOA)

From the point of today's businesses, a Service Oriented Architecture (SOA) can be said to be an architecture that will enable businesses to think about and organize their IT infrastructure in a manner that suits their customized processes. It is basically a collection of services which interact with each other and together will provide the necessary functionality for the required processes [1].

A service in this context can be defined as a software component that has a well defined interface and underlying operation. Services are generally used by clients without understanding of their underlying implementation. Simply said, they do not need to know 'how' the service works, just what it does and how it can be accessed. This separation of the service interface and the implementation is a vital point in an SOA. The following diagram, Figure 1, shows the general model used by an SOA. The service providers will register their services offered in public registries accessible to clients. This registry is then used by Service Consumers or clients to find services which match their requirements or criteria. If a matching service is found in the registry, then the consumer is given the information it needs to find and utilize that service.

As the services in an SOA are loosely coupled and highly interoperable, they may be developed using many different technologies. These services will of course be re-usable. These points give many advantages of using SOA in businesses. Because of platform independence, companies or organizations using an SOA can implement any software or hardware they require. It also improves flexibility, productivity and speed of deployment of functions for both the business and IT. It is also more adaptive and this will create monetary savings overtime for the business as well [2].

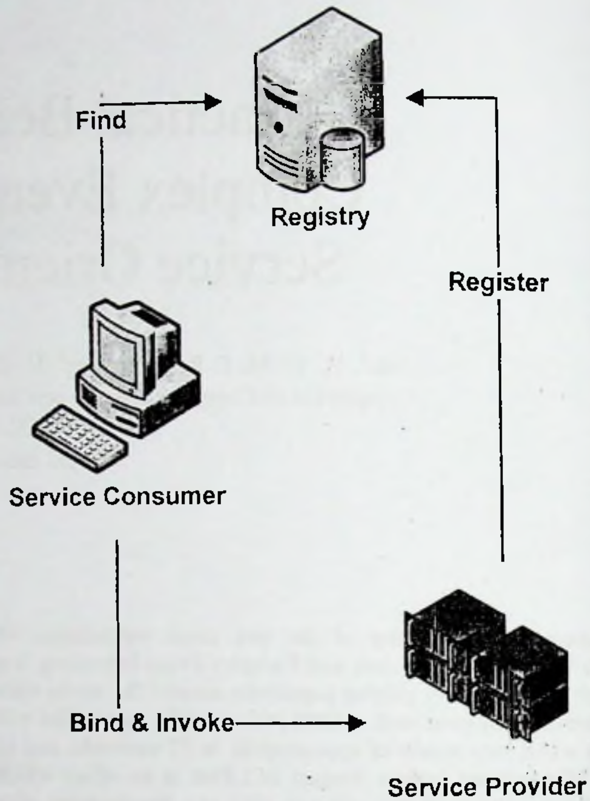


Fig. 1. SOA Structure.

B. Complex Event Processing (CEP)

The other major technology related to this paper is Complex Event Processing (CEP) is an offshoot of Classical Event Processing; where a multitude of events taken from an event cloud are examined for the purpose of identifying patterns and extracting usable knowledge based upon pre-defined criteria or conditions. Simply put, it is a means of extracting high level information from a complex array of situational bits of data.

As a simple example for a potential CEP application, consider a network management application which needs to function in a high speed environment with rules and conditions. With pre-defined patterns to look out for set within the system, it will be exposed to the network traffic event stream, where it will listen for and identify the necessary patterns and take some defined action set for that application.

At the heart of a CEP system will be some implementation of an Event-Driven component surrounded by architecture to process information and extract necessary knowledge. One architectural model is simplified in the above diagram; Figure 2; the Joint Directors of Laboratories (JDL) architecture for a CEP system [2]. This model highlights the key events that would take place in a CEP systems functioning; preparing and refining the event stream for processing, analyzing multiple event objects to pre-defined patterns and then carrying out the process changes or alerts.

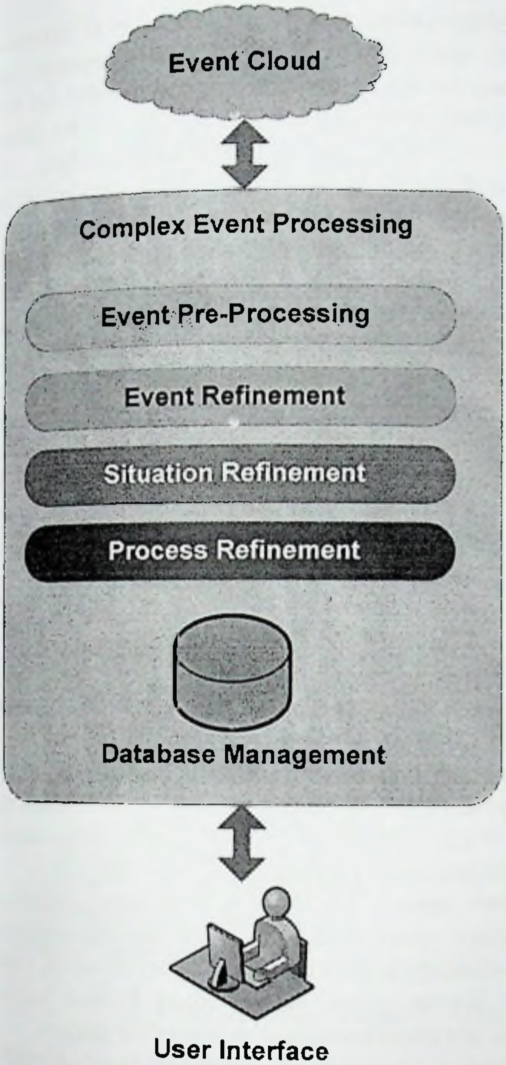


Fig. 2. CEP model

CEP is a technology increasingly used for building and managing information systems including:

1) Business Activity Monitoring
2) Business Process Management
3) Banking
4) Event-Driven Architectures
5) National Security

C. *Enterprise Service Bus*

When talking about SOA implementation, another vital concept that one needs to be familiar with is the Enterprise Service Bus (ESB). In simple words, an ESB can be said to be a message broker that provides a comprehensive message queuing and forwarding system, generally with support for many industry standard message specifications such as JMS or SOAP.

The figure below; Fig. 3; displays a few of the possible service endpoints of an ESB. A typical installation of an ESB in a corporate environment would likely also involve services such as Security (encryption and signing), BAM (Business Activity Monitoring), Message processing and mapping, Event Interpretation and Mediation.

In the general case, ESB's use XML as a standard communication language and provide support for web services. It also brings interoperability among different platforms and technologies; a key feature in implementing an SOA as well as from a business perspective, it is able to apply certain business rules and policies (e.g. in message routing).

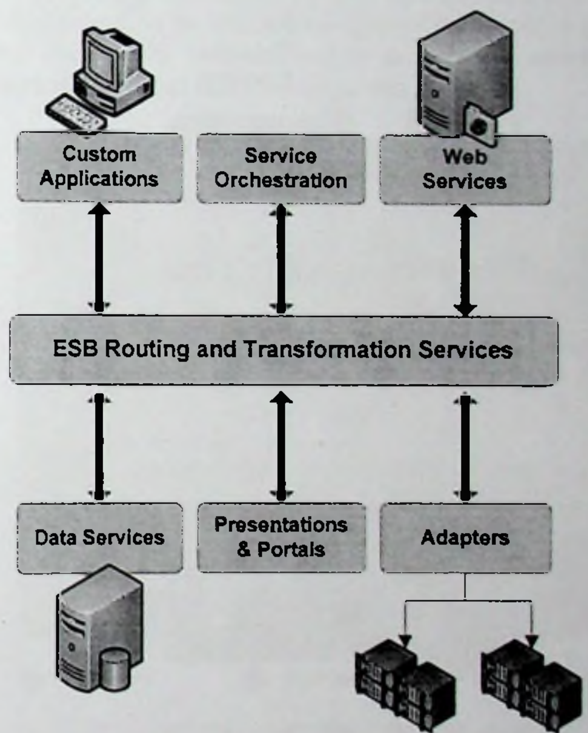


Fig. 3. ESB Services

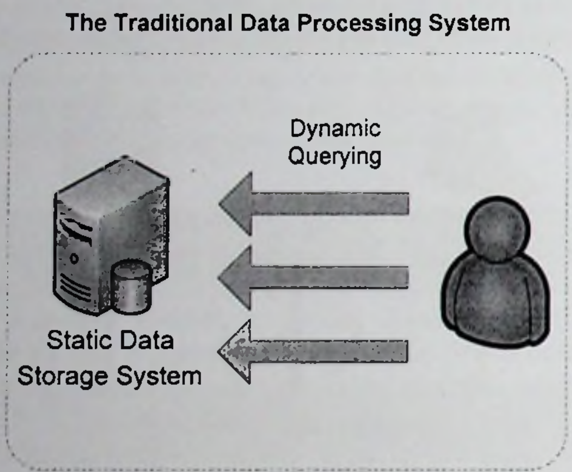
For a business, in the end, this translates into Security, reliability and availability in their business functionalities as well as cost effectiveness and flexibility in any restructuring or redefining of processes and business functions.

III. CEP ON A SOA

Now that we have taken a look at the different components individually, we will now look at the combining of these technologies, and the impact of this merging.

Firstly, we need to understand how a CEP system would traditionally be put in place. A CEP system with the functions defined as in the introduction would normally be pre-configured for a particular application or instance with pre-set rules and criterion for identifying necessary patterns in the data. It would also be used as a stand-alone application component used in association with a larger software system.

The combining of CEP into an SOA environment will somewhat be a u-turn to how normal data processing or querying applications would function. In a traditional database like system, the data will be in general static whereas the queries will be of dynamic nature.



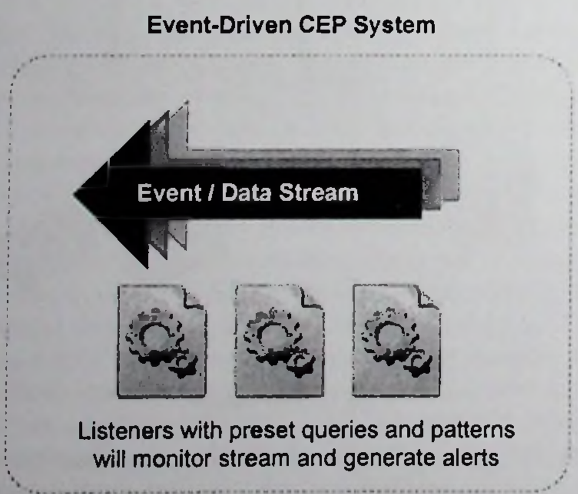


Fig. 4. Event-driven vs. Traditional

Now, the kind of environment we are interested in, which is that of an SOA, there will not be prominent static data storage of this manner and instead what we will have, is data streams rapidly flowing among different service or processes. There will be a multitude of different products which will be rapidly interacting over a network environment using services. If we are also looking to develop a system which will be more sensitive or re-active to data pattern fluctuations; which will of importance in implementing real world business decisions based on variations; then the system will also have to be of event-driven nature. When thinking about systems which will be of event-driven nature in general, this concept of static data and dynamic querying will be more or less inversed. The criteria which we will be using in order to monitor the data will more or less be static, but the data stream will be rapidly changing.

To sum up the advantage in a sentence: CEP enables adding real business intelligence to the rapidly changing, loosely coupled world of SOA. How this will happen is; first, SOA will create a highly dynamic event based system with rapid interactions and large amounts of data streaming among applications. Into this dynamic data stream, or the event stream generated through service interaction, we add the power of pattern identification and implementing real business intelligence through CEP. SOA creates events, and CEP brings in the intelligence in analyzing it. Do note that the amount of actual events we are talking about is simply huge *and* very rapidly changing and therein lies the real power of CEP.

IV. Introduction to SCI-Flex

This section is an introduction to our project into the world of enabling CEP in an SOA environment: Project SCI-Flex. SCI-Flex stands for Flexible Integration of SOA and CEP. Therefore the key effort of the SCI-Flex team was to create the integration of CEP with SOA, while maintain the scalability and flexibility that an SOA environment would offer.

As of now many CEP users are looking forward to integrating those scenarios in SOA environment though some of them are currently in the mission. SCI-Flex is indeed a part of a larger system where as the others involve various optimizations and improvements to existing CEP engines.

The Mediator that interconnects Esper and Apache Synapse, also known as the SCI-Flex Synapse Esper Mediator, lies at the heart of the implementation of the project. Being the core and the principal idea around which the entire project was developed, the mediator was perhaps the first component that we looked into and its crucial functionality was developed starting at the very onset of our project.

As the main infrastructure of SCI-Flex, we have the implementation of a Synapse Esper Mediator that was capable of reading a message that came into the ESB in XML format, forward it to the CEP system, and then obtain the response generated from the CEP, and forward it to the endpoint specified.

Our first focus was to add logging infrastructure to the initial prototype which we created. Next focus was to allow the user of the mediator to better make use of the Synapse Configuration in routine updates instead of resorting to changing code as an alternative. The next major portion was the Axiom Mediator, which now allows one to make it possible to make use of Axiom (Object Oriented XML) instead of pure-XML as means of communication between the ESB and the CEP engine. Based on the project skeleton defined for the initial work, we then moved to look at the registry integration process.

The key requirement for the registry arose in the process of EPL statement execution life cycle. But this will be bound to the mediator throughout the lifetime of the backend server if we do not make use of a registry. Hence with the registry integration the entire process becomes dynamic and able to be modified during the running time of the ESB as well as it enables the possibility of storing the query on a backend registry (which is a REST based resource repository) compatible with Synapse was then thought of. What happens here is depicted below in Figure 5.

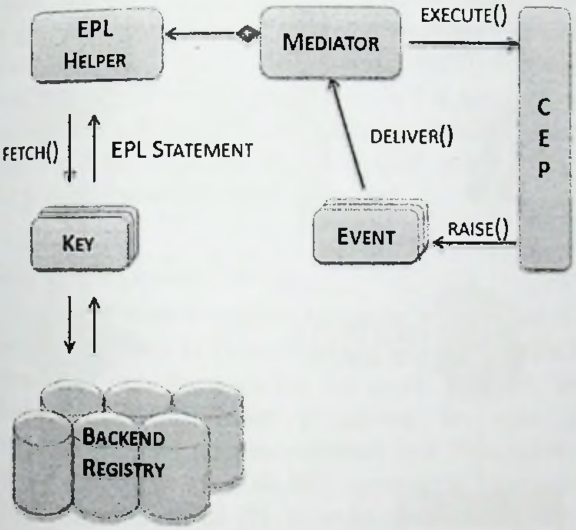


Fig. 5. The integration of Registry in SCI-Flex

By performing this integration, it was now possible to change the resource at the registry and have that same change being reflected immediately or at a specified delay. A similar procedure can be used to store the endpoint to which the response is to be delivered. As you can see in the figure 5, the EPL query helper is capable of handling dynamic modes of operation, by providing necessary abstraction of the EPL query to the underlying plug-in environment, and thereby it relieves the developers from having to differentiate the static and dynamic scenarios.

The next step was to think of the concurrency support of the mediator which would enable it to handle many transactions at the same time. In this case, the concurrency support available as a part of the JDK was also considered in getting the best out of the resources available to us towards making an excellent environment for concurrent operations. The Minimization of the use of common resources which in turn have various interdependencies as much as possible was found to be the only way to increase this concurrency support. In between we also had to make some critical assumptions based on which we have improved the flexibility of our implementation. For example, some assumptions would be that the Mediator operation is the most frequent operation and is the key deciding factor of the speed and performance of the mediator.

The modification of the query and updating of the mediator's execution accordingly is the most time consuming but usually not the most demanding operation in most scenarios. Thus, we have achieved the mutual exclusion of the two events, by making sure that mediation does not happen during the period of query modification.

The plug-in management infrastructure for the graphical user interface is the next major addition to our project. Each and every modification (configuring and monitoring) on the user interface to the mediator instance should operate over this infrastructure. As the need for management infrastructure grew with the need to keep the plug-in as it is, we added an additional layer that handles management features.

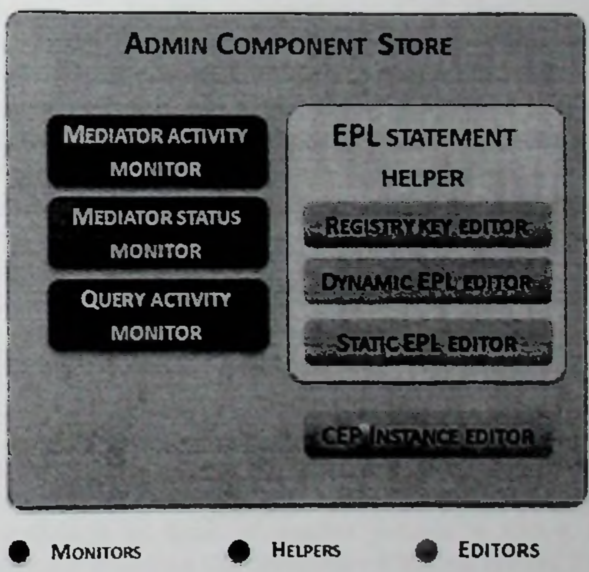


Fig. 6. Interacting Management Components in SCI-Flex

To achieve this, and also to maintain the flexibility of the design, the management (or administration) layer was added as a set of components. Each of these components will then be added to a container that will reside in the global configuration so that anyone who can access the global configuration can make use of the management facilities. There are three main types of management components named editors, helpers and monitors.

Each management components can be identified by an UUID (Universally Unique Identifier).Thereby; each mediator instance can have its own set of associated management components which improves performance as they share tasks among them and reduce the time overhead. The plug-in also introduces some core components and utilities as well, which are usable not only by the mediators but by any other related object(s). The Core components are further grouped into four parts which are named as Activities, Workflows, Administration and Statistics.

The administration of various components that are part of the ESB can be achieved in two ways according to the current implementation:

1) Java Management Extensions (JMX) based infrastructure
2) Web Services based infrastructure

Out of these two possibilities, the administration is available in its entirety as a WS based infrastructure. However, as of recent times the primary focus of the WSO2 ESB has been the WS based implementation, due to its higher capabilities. Moving in the direction of the popular choice, we have exposed all of the management through a single administration web service. The details of this administrative service are summed up in the figure below Fig. 7.
The benefits to use WS based approach.

1) Component level distinction
2) Language independence on client side
3) Low coupling between server and client
4) High flexibility and ability to make use of the powerful WS stack for added features (such as security)
5) Strong API based separation for server end
6) Ability to automatically generate client stubs based on the generated service contract.

The next major issue was to package our plug-in to suit to the ESB or any software component which could have a potential use for it. The entire plug-in can be used as separate jar archives, and also as a single OSGi bundle at which we all are focused as much as it's preservation capability as of backward compatibility over versioning.

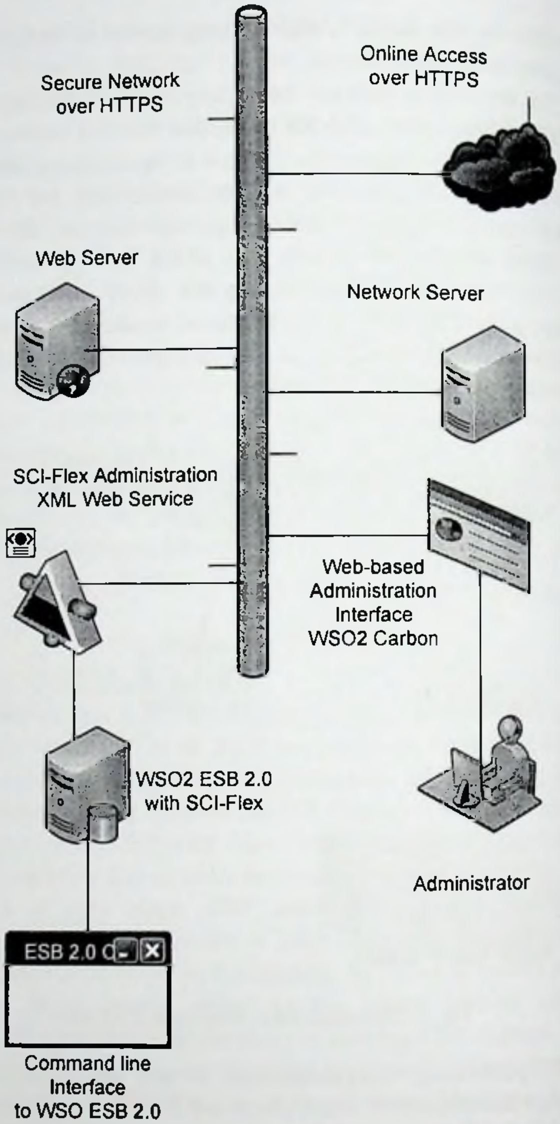


Fig. 7. SCI-Flex Administration Web Service

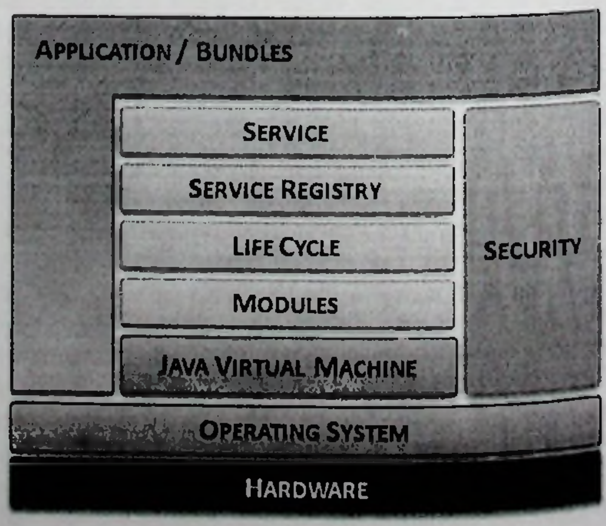


Fig. 8. OSGi Architecture

Looking at the above figure; Figure 8; the OSGi bundle we developed for the Synapse Esper plug-in is represented in the Green area, and it has the capability of implementing a robust modular architecture of hot deployable components unlike traditional jar archives in addition to various other features. We have used Apache Felix in creating the bundle along with Apache Maven.

V. Real World Applications

There is indeed an increasing gathering in the world of SOA fans towards the merging of SOA and CEP, simply because of the many practical usages this merging would bring about that would otherwise not have been a possibility. SCI-Flex too was created to be a real part of this movement, and we definitely envision a bright future for this venture with the potential for a multitude of real world possibilities. In this section therefore, we will look at just a few of these major possibilities for some real world implementations that a CEP would bring about while operating in an SOA environment.

A. Air Traffic Control

Air traffic control today is becoming a major focus of the aerospace industry in an environment where the traffic is ever increasing, and the demands for quick response, accurate forecasting and efficient processing are also gaining prominence. Based on data collected in a 2006 survey, on a typical day, more than 40,000 commercial flights operate within the US airspace [5]. In order to efficiently and safely route all this increasing air traffic, current implementations of traffic flow control rely on a centralized, hierarchical routing and processing strategies that performs flow projections with a considerable delay, maybe even ranging from one to six hours. One consequence of this is that the system as a whole will be slow to respond to developing weather or airport conditions and these minor inaccuracies may result eventually in large scale traffic congestions over larger regions [6]. As a solution to this, multi-agent systems are being looked into to be deployed across regions [7].

Now let us try to view this solution in the current terms of SOA and CEP and see how the merging of these concepts can be applied here to create a more optimal solution.

First, consider the operations of Air Traffic Control at a single installation. Even this itself, is a daunting task. It involves several major focuses including;

1) Radar coverage
2) Ground Control
3) Flight traffic mapping
4) Local or Air Control
5) Clearance delivery
6) Approach and terminal control

These individual components themselves deal with a large amount of data and would then employ some form of CEP in order to obtain information out of the raw data stream which could include sensory and situational data from many locations. Even to this single component, the addition of an SOA environment could result in increased range and flexibility as well as easier implementations across different systems. Merging the SOA and CEP operations in this case then would enable this.

Moving a step ahead, these components are obviously not meant for individual operation. For any kind of efficient and accurate traffic handling to occur, at the control point, we would need to bring together data from all these applications and then make our decisions based upon this aggregated data. Once again, implementing this inside an SOA would without a doubt cater to this need; and also, if we add to it the power of CEP, this whole scenario may be restructured to perform much faster and generate information out of raw data more efficiently. This would be because with the merging of CEP and the SOA environment, the processing could be performed on the many raw data streams in one go instead of it being done separately in different locations and the results simply being sent over to be first aggregated, and then analyzed to make a decision. With this, it is easy to see how this whole operation can be made to operate across a network of different installations thus providing the same processing power across a much larger region while maintaining the speed and accuracy. Note of course that added advantages such as flexibility and scalability will be there in utilizing SOA.

Note that the current systems are facing increasing amounts of stress in terms of the increasing air traffic and thus, the amounts of rapidly changing data they need to work with are also rapidly increasing. The need to make more accurate predictions on terms of weather patterns is also a rising demand. If we have CEP integration within the system, it is not too hard to deal with massive amounts of data, and with SOA implementation, the system may be scaled or restructured much easier.

B. National Security Monitoring System

Government and the commercial industries of a country increasingly require the integration of a complex mix of business activities and services from multiple public and private organizations to fulfill their missions and deliver important services. Usually there will be a multitude of programs to modernize computing, communication structures, and information services that will be aimed at helping these operations reach higher levels of performance through new business processes supported by advanced technology and information systems. Taking this a step forward to consider a multi-agency co-operative task across the whole country such as monitoring the Security situation of the country as a whole is much more of a daunting task, as it will require this same type of integration among agencies, businesses and public organizations in order to function properly [8][9].

Looking at some modern approaches to dealing with this requirement, we see that the features of such a system would be very complex. Obviously, we will be talking about a distributed system possibly spanning a large portion of a country. As a base these systems would include some of the following features.

1) Real-time vulnerability assessments as well as some vulnerability management using a combination of host-based, network, and passive vulnerability assessment technologies that would operate on a continual basis.
2) Critical security event monitoring real-time event aggregation as well as log analysis and normalization. Also, security event alerting using some means of statistical alerting that would be based upon the detection of network anomalies.
3) Compliance monitoring using a variety of agent-less configuration files that maybe designed for specific government rules and regulations. This will also include some sort of customized alerting mechanism, with customizable actions as well as event endpoints which would operate based upon recognition of preset patterns [10].

Note of course that all these features would need to operate on a vast distributed network which would be encompassing many different types of hardware infrastructure as well as software systems. Also, the need would be there to easily restructure or scale this system easily and often based on changing requirements or other factors such as change in government agency infrastructure. In this scenario, it is easy to see that the implementation of an SOA system would provide these advantages and more.

Looking into the application of CEP into this scenario, we can also see that the amounts of data that this type of system would have to handle will be tremendous. It would consist of a large number or parallel streams of sensory data from around the country as well as situational data from a vast multitude of sources. With this data properly aggregated, a proper CEP system would be able to efficiently analyze this data and draw meaningful conclusions from it. It would also respond quickly to situations; which is a vital factor in such an application. If this system was operating on an SOA, the aggregation of data from myriad sources would not be a very demanding task.

Therefore, in this scenario as well, the integration of CEP into the SOA environment would be an ideal methodology to solve the problem at hand.

C. Network Traffic Monitoring

Now we will consider a more generic example for CEP and SOA integration: the realm of Network Traffic Monitoring. Here, the suitability for an SOA is much more obvious. Talking about networks, we can easily expand our scope to include a wider network, or an internetwork which would encompass different hardware and different operating platforms and application software, all of which would at some level have to communicate with each other. Thus, interoperability becomes a key factor. Also in a network, scalability as well as incorporating structural changes without affecting other nodes is an important point. Along with all of these facts, business functions must be delivered across networks efficiently, and I could well be for a large audience if it is a public service of some kind. Implementing an SOA over this infrastructure grants obvious benefits. Now moving onto monitoring and analyzing traffic over such a network, once again the importance of SOA is seen as we would need to aggregate sensory data from multiple locations and depending on the network, work with several administrative points in different locations.

Integrating CEP into this environment will vastly increase the processing potential of a tool to monitor network traffic and allow users to create a system with considerable intelligence. Generic network traffic will be very rapidly changing, and depending on the size of the network, the amounts of information being passed around also would be substantial. Using CEP, it is easy to lay down predefined queries into listeners on this continuous flow of data with rules and regulations that will help identify certain patterns. For example, systems may be configured to take action when network traffic reaches a certain limit, or if users are accessing potentially unethical sites etc. Another advantage in utilizing CEP upon an SOA environment would be that these rules or queries themselves could be easily changed by adjusting the configurations on central networking devices.

VI. How SCI-Flex Can Help You

We will now observe how the SCI-Flex system and its method of enabling CEP in SOA will work in a real life scenario. We will be applying the concepts of this project to one of the scenarios we have discussed in the previous section.

Let us take a look at SCI-Flex in operation in the realms of network traffic monitoring. In this example, we will consider a simple network traffic monitoring tool that will be powered by SCI-Flex. The image below; Figure 8; outlines the structure and the flow of this application.

The situation we are looking at is monitoring traffic in a network by analyzing a stream of network traffic and identifying patterns in the data and then generating suitable events to suit this pattern.

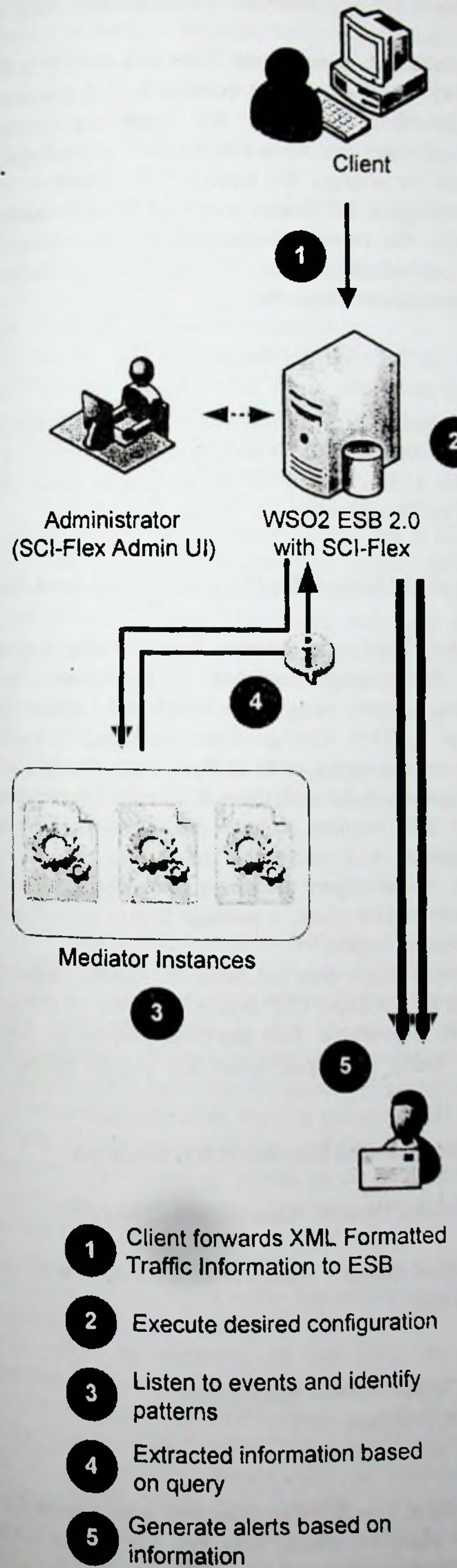


Fig. 9. Network Traffic Monitoring Tool

Note that with the power of CEP, this amount of data we can work with is huge as well as it can be very rapidly changing. The whole situation with its rules and patterns is also highly customizable and easily adjustable. The WSO2 ESB sits at the center of the application, enabling the SOA environment which is the infrastructure on which this whole system with CEP will operate.

Moving on to the actual application; firstly, we will be using a Client that will be plugged into the traffic stream which will then be forwarded to the WSO2 Synapse ESB which will have an embedded instance of SCI-Flex running inside it. This message may be of any valid transport such as JMS or HTTP. This client can be a small Java application that is capable of sending XML messages to the Apache Synapse ESB.

When the ESB receives this information, it will understand from where the information arrived and based on that, it will execute the suitable configuration, which will in fact forward the incoming message stream to the Mediator and through it to the CEP system. This suitable Synapse Configuration will be automatically loaded and it will have defined with it a set of pre-defined mediators that will be listening in to data to identify some pattern.

These mediators will share the same CEP instance but will be executing different queries that can be pre-defined based on the system requirement. Each mediator also may or may not share the endpoint to which it listens with another mediator. This will make the whole application well suited for an environment which will be able to facilitate multiple inputs as well as different filters depending on the type of the input and its source. These mediators are capable of filtering out useful data from the raw event stream, and there by extract information out of the raw data, and feed it to the ESB in a format that it understands. The Mediators are powered by a pseudo listener that is capable of converting the data into information just before it reaches the ESB.

Our system will enable these mediator instances to be configured with customized actions to be taken in the case of a certain pattern occurring in the data. If this pattern is recognized in the data, the listener will generate the suitable notification alert. By default, SCI-Flex will allow this alert event to any type of message that is supported by the Apache Synapse based ESB. If the user requires a custom message, this can be separately implemented for this purpose and the system configured to handle it. The endpoint of this alert can also be customized and it can be sent out to any location that the user may prefer. To put it in one line, the ESB which is now aware of the information it receives and also aware of where it should forward it to will carry out the process of generating alerts.

This whole process of the application may be explained in the five steps shown in the diagram. There will be several sub-steps which will perform these overall feats, but we will not detail them here. Do note that the entire process will be powered by the SCI-Flex mediator which will provide the merging between the Synapse ESB and the Esper CEP system.

VII. Future Work

The realm of CEP in a SOA is still a new one. In spite of the developments and achievements highlighted in this research paper, there are many improvements and additions to existing systems which will enable even greater advantages in this field. We will note a very few of them here;

A. Web Service Eventing Integration

Web Services Eventing (WS-Eventing) will provide a protocol that will allow Web services to subscribe to or accept subscriptions for event notification messages. WS-Eventing defines a protocol for one Web service called an "event sink" to register its interest called a "subscription" together with another Web service which would be the "event source" in receiving messages about events. These events are generally known as "notifications".

Web Services today have become a very popular means of implementing an SOA. Therefore, if we are to consider the future of CEP systems working together with SOA, it is obvious that these merged systems will need to have WS-Eventing integration in order to be a popular choice.

Also as CEP with SOA systems would in general be working with large event streams, it would be important to have a standard protocol in place handling the event subscriptions among the different and varied sources and endpoints in the system in order to ensure that the system performs efficiently and accurately.

B. Web Services Notification

A common practice that is increasingly used in Web Services is a Notification Model which is simply a model of a service provider initializing communication with a service requestor based upon certain subscriptions. WS-Notification is a standard approach to Notification that defines different aspects of this subscription model such as standard messages, operational requirements and XML models which describe topics. Once again, in order to ensure increasing integration of web service based SOA systems with CEP operation, it would be important to have a proper subscription model running the show.

C. Web Services Topics

Web Service Topics or WS-Topics are used as a part of WS-Notifications to organize and categorize items of interest for subscription. The WS-Topics specification defines three topic expression dialects that can be used as subscription expressions in subscribe request messages as well as other parts of the WS-Notification system. It also outlines an XML model for describing metadata associated with these topics [13].

A potential future enhancement in the area of CEP in SOA is to use WS-Topics to make it possible to define and address topics (event notifications), and organizing topics into hierarchical structures, known as Topic Trees and specifying synonyms (or aliases) for topics. This relates to actually implementing the WS-Topics portion of WS-Notification as a blend to the WS-Eventing implementation. This would thereby provide an excellent basis for a solid WS-Notification implementation on a later date.

VIII. Related Work

In this section we will take a look at interest in the field of merging SOA and CEP by looking at certain products which also cater to this market as well as new research which is heading up the same alley.

A. Comparison between Sci-Flex and Apache Camel Esper Plug-In

Apache Camel is a criterion based routing system with support for message mediation. It implements enterprise integration patterns using either Java based Domain Specific Language or XML Configurations, by which it implements routing and messaging rules of the system. In the traditional sense, Camel can be said to an ESB as it implements many standard ESB services such as intelligent routing, message transformation and mediation, monitoring as well as data services. Its developers view it as being able to operate as a smart web service client, a message broker as well a routing and mediation engine.

It is interesting to note that under the Apache Camel project, a plug-in for the Esper CEP system that we have mentioned in this paper is available. This and other plug-ins for Camel are actually being developed under the project called 'Camel Extra'.

Uniform Resource Identifier (URI) format [4]:

esper:name[?option1=value[&option2=value2]]

You must specify a pattern or an eql statement to query the event stream.

e.g.

```
from("esper://cheese?pattern=every
event=MyEvent(bar=5)").
    to("activemq:Foo");
```

Looking at how SCI-Flex compares with Apache Camel; in the first place we should note that configuration and EPL statement setting in Camel-Esper-Plug-in is not that easy as in SCI-Flex to process events. Camel requires a separate java class in the case of setting queries and configuration parameters to perform significant event identification over a stream of events. But in SCI-flex it is pretty easy to handle those scenarios as the system is totally depending on XML. Esper configuration and EPL Statement setting in SCI-Flex is handled through XML sample documents, thereby reducing the interdependency as well.

For the same reason, configurations in Camel will be static whereas in SCI-Flex, different configurations maybe loaded into the ESB at runtime. Also, writing applications for SCI-Flex also becomes much easier compared with Camel, where one would need additional dependencies.

SCI-Flex also provides a comprehensive UI for a user to fully monitor and manage the whole mediation system by observing mediator activity, load, query statistics and more. This manner of versatile UI is nonexistent on Camel. Camel is also not capable of monitoring statistics real time.

Looking at actual test results, The time taken to process over Camel-Esper plug-in is almost 2135ms but in SCI-Flex it takes almost 18.3ms to mediate over XML Mediator and 7.4ms over Axiom Mediator. Obviously this is a massive difference in performance. Mostly this extra time on Camel will be due to the necessary set up of the ActiveMQ broker. This is an overhead SCI-Flex does not have. Note also that unlike in Camel, SCI-Flex does not have to recompile the Java files after changing the query.

In conclusion therefore, SCI-Flex clearly stands ahead in a comparison of mediation capability and performance with the Camel Esper Plug-in.

B. IBM Research

It is also noteworthy to mention here the research which is taking place at IBM regarding SOA and CEP. Extending the capabilities of SOA into a whole new level has been a focus of IBM for quite some time as can be seen by the many developments and ventures they have taken in this area.

One can specially note that the amount of research and the projects extending into SOA itself is massive. IBM is a key player in the world promoting the use of SOA as well as the development of methodology in making the transition to SOA easy as possible for a potential business: a term which they coin as "Smart SOA" [11].

One of the main focuses of this research is optimizing business event processing or rather Business event processing to support business objectives.

A large effort is underway at the IBM development communities in order to create real time CEP engines. One such initiative aims to use CEP technology embodied in their rule-based engine of IBM Active Middleware Technology and extend it towards the development of real-time CEP applications. Their proposition includes a framework that includes an integrated development environment (IDE) for defining rules and regulations, and one that if given a set of rules will generate code for a CEP application and enables the users to determine the time constraints set on the response times of particular applications towards a set of considered events [12].

REFERENCES

[1] Aziz S., "SOA: Look Beyond the Myths to Succeed", (2006, March).[Online]. Available: http://www.infosys.com/IT-services/systems-integration/white-papers/soa-look-beyond-myths-succeed.pdf

[2] Aziz S., Banerjee J., "SOA: The Missing Link Between Enterprise Architecture & Solution Architecture", SETLabs Briefings, Vol5, No.2, March 2007.

[3] Bowman C., Llinas J., Rogova G., Steinberg A. White F., "Revisiting the JDL data fusion model II (2004)" In Schubert J. and Svensson P. (Eds.), Proceedings of the Seventh International Conference on Information Fusion – 2004

[4] Apache Camel Esper plugin - http://camel.apache.org/esper.html

[5] Chatterji G. B., Soni T., Sridhar B., "Aggregate flow model for air-traffic management". Journal of Guidance, Control, and Dynamics, 29(4):992–997, 2006.

[6] Agogino A. Tumer K., "Distributed Agent-Based Air Traffic Flow Management" by Oregon State University, AAMAS'07 May 14–18 2007, Honolulu, Hawai'i, USA.

[7] Menon P. K., Sridhar B., Sweriduk G. D., "Optimal Strategies for Free-Flight Air Traffic Conflict Resolution", Journal of Guidance, Control, and Dynamics (1999).

[8] Hoffman K. C., Pawlowski T. J., Payne D. L., Zheng K., "Enterprise Business, Computing, and Information Services in a Multi-Agency Environment: A Case Study in Enterprise Architect-Engineering", Proceedings for Middleware for Web Services (MWS) – 2005

[9] Hoffman K. C., Knouss C. D., Pawlowski T. J., Zheng K., "A Service-Oriented Architecture in a Multi-Agency Environment: A Case Study in Enterprise Dynamics", 10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'06) – 2006

[10] Deraison R., Gula R., Ranum M., "Unified Security Monitoring (USM) Real-Time Situational Awareness of Network Vulnerabilities, Events and Configurations". September 5, 2007.

[11] IBM Smart SOA Approach - http://www-1.ibm.com/software/solutions/soa/launch/

[12] Adi A., Barnea M., Botzer D., Magid Y., Oren D., Rabinovich E., Shulman B., "Generating real-time complex event-processing applications"; IBM SYSTEMS JOURNAL, VOL 47, NO 2, 2008

[13] Niblett P., Graham S., Vambenepe W., "Web Services Topics 1.3 (WS-Topics)" OASIS Standard, 1 October 2006