Integration of Enterprise Applications

Research Project

Submitted By:

Team - Enteurs

Name	Student ID	Mobile	Email
Anuragh Reddy Gade	007224308	925-406-4034	anuraggade@gmail.com
Anirudh Mittal	006491563	408-449-1849	Anirudh.mittal89@gmail.com
Gowthami Chekkara	000165478	408-416-3638	School.gow@gmail.com

Submitted To:

Prof. Rakesh Ranjan

Submission Date:

05/09/2011

Abstract

In this paper we propose a solution to integrate Immigration related Enterprise applications of various countries to communicate with each other. For this communication to take place we need to provide a rule based routing and mediation engine using Enterprise integration patterns. We have used an open source implementation of Enterprise integration patterns which provides a basic framework for various enterprise applications to communicate with each other with type-safe smart completion of routing rules in your IDE using regular Java code without huge amounts of XML configuration files. In this paper we present the existing techniques and also compare it with our solution which makes the integration among the enterprsie applications in general and in specific for our enterprise application feasible for a developer without any concern to the mediation rules, routing and implementing the enterprise integration patterns.

I.INTRODUCTION

Various communication models are used to implement the communication between the Enterprise applications using some mediation rules and with the implementation of Enterprise Integration Patterns. Enterprise Integration Patterns (EIP) represents possible design solutions that may be used to make the communication between enterprise Applications possible. Leveraging the power of Internet and building on XML permit systems to share information provide communication and invoke other system functions irrespective of their programming language, operating system or hardware platform. Developing integration solutions, however, remains a difficult task for the designers, who may leverage design knowledge available in the form of Enterprise Integration Patterns. Conversations among Enterprise applications, properly designed, can guarantee consistent and reliable execution of integrated systems .long running conversations can be supported through available communication protocols like SOAP[1].

Message routing is the foremost functionality for an Enterprise application as it provides the fundamental capacity to support intelligent interconnectivity between service requestors (or consumers) and service providers, and by using it as the communication backbone, the interactive topology between services can be simply point to point, and also multi-point to multi-point in complex situations. Since the invocations are independent of the service location and the involved protocols, they can be dynamically located and invoked, which is important for communication in enterprise applications. Existing design approaches for application integration neither provide mechanisms for designers to select appropriate EIP for integration problems nor do they provide mechanisms to generate appropriate integration model. Even if the applications are integrated using the EIP in future it becomes a problem when considering the scalability of the application to use different design patterns, in such a case the entire conversational model has to be changed which becomes a heavy tedious task for the developer and in terms of cost as well[1].

For the integration to be carried out with enterprise applications we are proposing a solution that takes care of the routing mechanism and mediation rules. We would explain the integration by considering the integration of the immigration system of various countries and the patterns required for the integration along with the open source integration framework called Apache Camel which takes care of the Mediation rules, routing(like JMS,Message Queues etc) and implementation of the Enterprise patterns required for the applications to be

integrated. We put forth the case study in which we have used Apache camel as an open source integration framework that used spring web services to provide platform independence [1].

2. BACKGROUND

2.1 Review of the present Integration Solutions

Firstly, Web Sphere Integration Developer developed by IBM, enables designers to wrap individual Systems as web services and choreograph them into business processes. The tool provides an integrated development environment to design and deploy integrated systems. It does not, however, suggest any integration strategies, and includes several dependencies on proprietary servers and modeling tools [1].

Secondly, iWay service-oriented adapters developed by iWay software, enable the designers to develop web services as adapters for ERP software's (such as Microsoft, Oracle, SAP, BEA, IBM, Siebel,), and join them into an integrated business process. It does not suggest any integration Strategies either [1].

Thirdly, Artix, developed by IONA Technologies, enables designers to develop web services adapters for legacy systems and integrate them using a hub and spoke approach. It claims to provide flexible and incremental integration approach (which may be considered an integration strategy) but does not extend support for developing conversation policies among the Enterprise applications [1].

Other Solution, BEA Aqua Logic Service Bus, designed by BEA, provides a service-oriented integration infrastructure which is based on web services techniques. BEA claims that this enabled the designers to design, integrate, deploy and manage systems exposed as web services. The tool, instead, does not propose any specific integration strategies nor does it support conversation policies among services as an implementation mechanism for the integration of enterprise software applications [1].

The above review of the present integration solutions for the integration of enterprise applications reveal that none of the commercial solution providers/open source providers provide any actual integration strategies. None of the provides explicitly support for the implementation mode we suggest in the application integration. The review highlights the essential need for tool that is independent in using the approaches to *designing* integration solutions. Integration designers need a tool that gives assistance to guide them with selection of appropriate integration strategies for a given task, and support generation of implementation mechanisms such as conversation policies for the implementation of various communication models that utilize the enterprise integration strategies for the integration of the enterprise applications [1] [2].

2.2 Role of Enterprise Integration Patterns (EIP) in the integration process.

Patterns play an important role in analysis and design and development of any software systems. A pattern is domain independent concept of common design structure that is proposed for recurring design problem. Each pattern describes when it can be applied, its design constraints, consequences, trade-offs of applying it in a particular context of developing software applications. A pattern does not provide complex code implementation, but rather help designers in describing and communicating design problems and solutions in a particular context of the integration process in specific or in the development process of the application in general [1][2].

When related patterns are framed together they form a “pattern language” which captures the relationship between solutions and problems in the specific context. Patterns and pattern languages together guide the designers through several decisions during software development and thus help the developers resolve the issue of conflicts in the specific context [1] [2].

In the domain of Enterprise Integration, Hohpe and Woolf have developed patterns that capture recurring design solutions for integration problems in real-time. These patterns are abstract i.e. they do not provide any implementation code or snippets in the integration process Instead, they provide solutions that help designers map integration problems against various possible integration strategies. They provide a real time match with the web services platform because they follow a message-oriented integration approach [1] [2].

All the Enterprise Integration patterns use the support of the messaging system.Hohpe and Woolf suggest 65 Enterprise Integration Patterns (EIP) organized into the following several categories: integration styles, channel patterns, message construction Patterns, endpoint patterns, routing patterns, transformation patterns, and system management patterns. No methodologies exist, however, to guide designers with selection of pattern(s) and construction of a solution for a given problem during the integration of the enterprise applications. All the patterns support the developers to resolve the conflicts in the context of integration [1].

2.3 DSL Route: Role in the context of Enterprise Integration.

DSL route provides fluent and best route definition. With DSL routes, the integration solution for the enterprise applications in specific and software applications in general are more agile and configurable since enterprise integration patterns (EIP) are naturally and in real time supported in DSL route model.Enterprsie Integration has to use the messaging system that provide important functionalities for reliable message delivery and complicated service routing’s Route proposes to improve the current messaging system [2] [1].

Messaging system have been widely used in SOA infrastructure since it can make applications more loosely coupled by asynchronous communication, which also makes the connection reliable as the applications need not have to be running at the same time[1].

Enterprise Service Bus (ESB) is introduced as the infrastructure to build and deploy the highly distributable applications and integration of those applications which is indeed a backbone for a SOA.It provides the integration platform that combines messaging system, data Transformation, service, and intelligent routing, and provides reliable efficient connection and coordination between diverse cross platform services. ESB provides four major functionalities: event handling, message transformation, protocol mediation and message routing between service provider and customer and have been reputed as the next generation integration software which is used in the SOA as well as for the integration of enterprise applications[1][2].

Enterprise integration is inherently and widely complex when applying to heterogeneous platforms, integrating diverse software applications and complicated business processes. One of the major stumbling blocks is the lack of a common pattern and knowledge of asynchronous messaging architecture used to build Enterprise integration solutions. All the 65 Integration patterns suggested by Hohpe and Woolf act as great resource in the integration process[1][2].

2.3.1 DSL content based routing module

Most of the Enterprise Software products support the dynamic route path by using content-based routing (CBR), which is

Known to be an intelligent routing. CBR enables the routing path to be changed at runtime depending on the message content to be communicated. XML is the intermediate message that provides the mediation between diverse data structures and formats as the messages are communicated between various enterprise applications in general and in specific to our application. In CBR, ESB determines the path the message to be routed by examining the message content. The routing should be determined based on a number of criteria, such as the specific field value, message type, existence of fields. XPATH is the language for exploring the XML document[1][2].

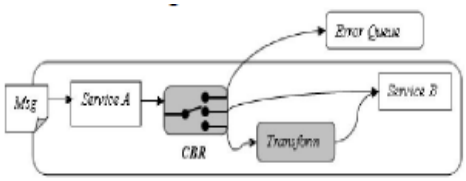


Fig 1: DSL-CBR on ESB

ESB routes the message to Service a in Step 1, and then gives the message out of Service a using a CBR module, which dispatches the message to three destinations: Service B, Error Queue or Message Transformation module before Service B. The ESB configuration file is an XML document which describes all the connections among the service providing components of the SOA application. During execution, the ESB routes the messages[1][2]

Exactly as specified according to the information in the configuration file.CBR is a frequently used routing mechanism which belongs to predictive routing. CBR has knowledge of all the possible recipients. As recipients are added, removed, or changed, the Content-Based Router has to be updated every time. Therefore, by using CBR alone can't supply an expected solution for varying diverse integration scenarios of the enterprise applications[2].

Domain specific language is a specification language dedicated to a particular problem domain and we use it to represent the integration patterns and replace the XML configuration in the integration process. Most of Enterprise Integration Patterns are abstracted into a function in DSL model. For example, we use "from" and "to" to specify the source and destination of the messages among various applications as a part of messaging process[2].

```
from("jms:quque:CBR.in")
    .filter(header("needTransform").isEqualTo("true"))
        .to("jms:queue:CBR.transform");
from("jms:quque:CBR.in")
    .filter(header("needTransform").isEqualTo("false"))
        .to("jms:queue:CBR.request");
errorHandler(deadLetterChannel("jms:queue:CBR.error")
            .maximumRedeliveries(3));
```

Fig 2: DSL based configuration for the CBR module shown in Figure 1.

In this paper DSL has been utilized as a part of the enterprise integration process using the open source integration framework called apache camel, which provides much easier ways to build and reconfigure the message routing path in our integration works.

3. EIP SAMPLE

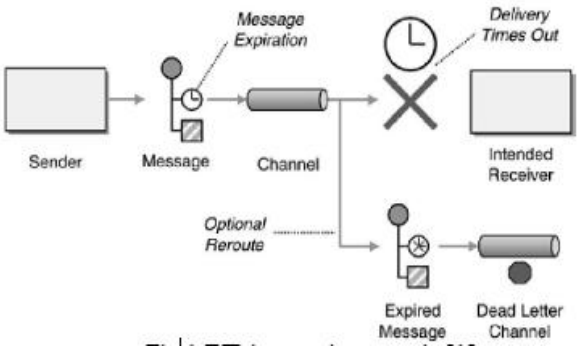


Fig: 3 Sample Process of Enterprise Integration Pattern

In the above sample, when producing a message, the sender may set an expiration time for it and then route it into a particular Message Channel as in the above figure. Message Channel is a basic component in EIP definition, where one application writes information containing the message to be communicated to the channel and the other one reads that information from the channel. It will try to deliver it to the intended recipient. This delivery operation may execute for several times because the receiver may not be online when the message has arrived or the message may expire before successful delivery, so a Dead Letter Channel is prepared to deal with all the non arrived message to ensure that the message can be managed in a reliable process preventing the loss of communication during the enterprise integration process. The above figure shows the corresponding patterns in the description [1][2].

3.1 Integration Problem Specified in this paper

Introduction

In this paper we have taken the integration problem corresponding to the immigration systems of various countries. The immigration systems have to mutually communicate with each other to verify the authenticity of a person entering the country, also there is a very need to pull up the background information corresponding to the person immigrating to a different country. During the verification process we need a sort of communication or enterprise integration pattern implemented to take care of routing the messages and mediation logic. We propose a solution to the integration mechanism by implementing the enterprise integration patterns related to the integration of immigration systems of various countries in agreement.

Need for Integration of Distributed Immigration Systems. In this we have the need to integrate diverse immigration systems to verify the authenticity of the people immigrating to other countries. Let us consider the example of American 09/11 attacks. If American government had the chance of communicating with the immigration systems of various counties and pulled up the background information of the people travelling in the plane then it would have certainly prevented the attack on WTC.

The reason why there is a delay in the communication of pulling up the information is due to lack of proper integration between the immigration systems that provide the criminal background of the people from the country where they are travelling.

3.1.1 Existing and proposed System

In the existing system if the immigration officer has to verify the authenticity of the person travelling or immigrating to their country they need to manually put a request to the country where the person belong to and thus have to wait for the response from them. This turned out to be a major drawback to the countries effected by terrorism where in they could not proceed on any action against the people they suspect. This also turned out a great hassle on those who are innocent because the immigration officers may not release or set free the people who are in suspicion.

The proposed system is in such a way that the immigration systems of various countries automatically pull up the information while the person is immigrating to a different country providing the complete and confidential information to the officers in charge of the immigration system making them to take up wise decisions. There might be some legal issues as well that can be negotiated with some agreements between

countries and thus considering the safety as a priority.

3.1.2 Hurdles in the Proposed System

The Integration of the Immigration systems is considered to be a complex process because different countries have a different implementation of the immigration systems. This implementation varies from the simple implementations to complex solutions. These are considered to be Enterprise solutions as they have wide range of use for protecting the security of the country at large and having the background information of the citizens of the country in specific.

To integrate these systems at large we need to propose and verify several enterprise integration patterns and provide routing and mediation logic implemented to integrate all the systems. We also need to consider the scalability of the systems as when a country changes its implementation of the immigration policies then they need to be reflected in the integration process as well.

To take care of these routing, mediation logic, implementation of enterprise integration patterns we need a concrete tool that would provide the integration process, provide the mediation logic and take control of the routing mechanism thus providing the implementation of enterprise integration patterns. We propose a solution to this integration process by using a open source framework called Apache camel as an Integration tool that implements the Enterprise integration patterns specified in the integration process for all the Immigration systems.

3.2 Role of Apache Camel in the Integration Process

What Apache camel is

Many companies use different applications within the organization or outside the organization. For example, their client might be having different application than their organization setup. Thus, many companies spend a lot of time and resources to figure out how will they be able to communicate with the other applications, and if they are successfully able to communicate with the application, will the information be rendered in the way it should be rendered. Thus, many companies forget to spend time on the business tier of their application and end up spending more time in setting up the connectivity between different applications. Thus, Apache came up with a product called Camel that solves this integration problem and helps the organization to spend more resources on their business logic.

The name Camel represents transfer of data from one application to another, as known that Camel helps in transferring of goods from one place to another. Thus, camel is a basically a routing library that uses various integration patterns that helps in transferring of information from one application to another application. Since, it a routing library, it is a very light weighted and does not need any installation and can be easily plugged in. The Apache Camel integration framework is based on java programming language.

Commonly the enterprise integration pattern that is implemented to communicate from one application to another is filter integration pattern, Filter patter is very easy to use and can be commonly read as English text. For example, if you need to transfer message between ActiveMQ and Web sphere, one can use the camel integration patter. Moreover, one can also make a DSL content-based router. Also, the apache camel is compatible with various programming language i.e. java, xml, and many more.

Let us have a look at the routing specified using XML content.

```
<camelContext>
 <route>
  <from uri="activemq:NewOrders"/>
  <choice>
   <when>
    <xpath>/order/product = 'widget'</xpath>
    <to uri="activemq:Orders.Widgets"/>
   </when>
   <otherwise>
    <to uri="activemq:Orders.Gadgets"/>
   </otherwise>
  </choice>
 </route>
</camelContext>
```

And the Java Implementation of the above code is as below.

```
from("activemq:NewOrders")
  .choice()
   .when().xpath("/order/product = 'widget'")
    .to("activemq:Orders.Widget")
   .otherwise()
    .to("acitvemq:Orders.Gadget");
```

Problems Solved by Apache Camel

1. It eliminates the need of knowing the deep knowledge of different applications and how to transfer and receive information from one application to another.
2. Also apache camel helps in solving integration problems within the application with the help of routing libraries.
3. Integration of components is made possible with the help of routing logic and mediation rules implementation using Apache open source framework.

3.4 The Problem statement which is given an implementation in the research project.

Problem Scope

According to the Mark Suster, in an article *Social networking Past, Present, Future*, he talks about the future of social networking where the Face book will be linked with Twitter and other social networking sites. Thus, we would like to propose a suggestion in terms of the users perspective that can someone build an application in which if we upload my pictures at Face book, it can automatically uploaded on Twitter and other social networking sites. To build an application, some of the things that should be known to the software engineer are[8][5]

1. The Integration approaches other social networking sites are using.
2. The deep knowledge of those applications on how they communicate with the external API's.
3. To come up with such a solution that there is a solid implementation for the Enterprise integration patterns being used in the Integration process.

Thus, in this paper we will be talking about how one can develop software that will help the user to upload the pictures from one social networking site to all other social networking sites automatically without having the hurdles mentioned above[8].

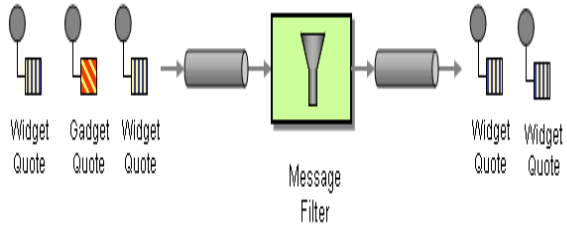
EIP'S in the Integration Process

First, let us look at some of the enterprise integration patterns that will help in transferring files from application to another. The integration patterns that will be used to solve the problem states are a routing pattern, a message filter pattern. Either message filter pattern or content-based

router pattern can be used to transfer information. Lets us briefly look at the integration patterns[8][7].

Message Filter Router Integration Pattern

The message filter pattern helps in filtering the incoming messages so that we can detect whether the message that was sent is the right message that needs to be sent. Basically, the message filter works as that the software engineer provides some rules that the incoming message needs to satisfy in order to pass through the filter. If the conditions are not met, then the message is not transferred ahead of the filter[9][6].



The apache Camel provides so much flexibility that one can write the rules in any programming language such as Xml, spring, Scala, Java, and many more [9].

So lets us write an example where we only accept messages in which the header “foo” contains a specific value “bar” [9].

Code in Java looks like

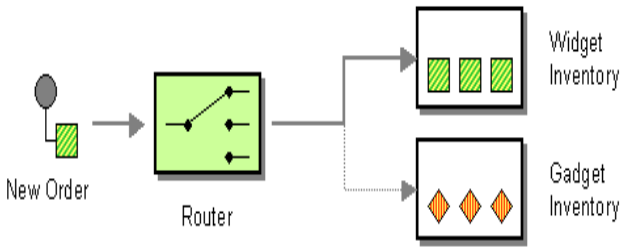
```
from ("direct: Point").

//from where the message is coming

filter (header ("foo").isEqualTo ("bar"))

//check where header foo has a value bar

to ("mock: Point");

//if passes send it to the specific location
```

Whereas the same rule if written in Spring looks like[9]

```
<camelContext
errorHandlerRef="errorHandler"
xmlns="http://camel.apache.org/schema/spri
ng">

 <route>

      <from uri="direct: Spoint"/>

      <filter>

         <xpath>$foo = 'bar'</xpath>

         <to uri="mock: fPoint"/>

      </filter>

   </route>

</camelContext>
```

DSL Content-Based Router Integration Pattern Using Apache camel

The content based routing integration pattern helps in routing the incoming message to its appropriate location by checking the content of the incoming message[9].



For example if we need to route the incoming message depending on the header “foo” value then the rule when written in Java look like [9]

```
from("starting:a")

.choice()

.when(header("foo").isEqualTo("bar"))

 .to("final:b")

 .when(header("foo").isEqualTo("cheese"))

  .to("final:c")

.otherwise()

  .to("final:d");

  }
```

When written in Spring XML it looks like this[9]

```
<camelContext
errorHandlerRef="errorHandler"
xmlns="http://camel.apache.org/schema/spri
ng"><route>

 <from uri="starting:a"/>

<choice>

<when>

 <xpath>$foo = 'bar'</xpath>

      <to uri="final:b"/>

      </when>

 <when>

  <xpath>$foo = 'cheese'</xpath>

  <to uri="final:c"/>

  </when>

<otherwise>

<to uri="final:d"/>

 </otherwise>

  </choice></route></camelContext>
```

After looking at the integration pattern, the other problem that the Apache Camel helps is that it resolves the integration with other application easier. To do so, Apache Camel uses a DSL (Domain Specific Language) that also can be written in any language such as XML, spring, Java, Scala, and many more. This helps in creating the entire route for the packet. One can easily write a route in java in one line [9]i.e.

```
from("seda:a").to("seda:b");
```

This statement routes the message from seda:a to seda:b.

After creating a particular route for the message and setting up the filtration of these messages using the integration pattern, now lets us focus on how to open an ftp server using Apache Camel[9].

One can easily write an ftp connection when written look like below [9]

```
<ftpserver:server id="ftpServer" max-
logins="10 anon-enabled="true" max-anon-
logins="5" max-login-failures="3"login-
failure-delay="20">

<ftpserver:listeners>

<ftpserver:nio-listener name="default"
port="3333" local-address="localhost"/>

      </ftpserver:listeners>

      <ftpserver:file-user-manager
file="classpath:users.properties" encrypt-
passwords="clear" />

   </ftpserver:server>
```

3.3 Related Case Study

Bond Trading System

Introduction

Abstraction of reusable form of idea is called Pattern[4]. In this case study we explain how patterns are used to solve real world problems. We used discovery process [4] to solve problems in Bond trading system. Patterns are used in various stage of project. It was initially used to choose the right pattern, how to combine and adjust patterns [4] to get best end results. The factors that affect our decision-making are system integration, client decisions, business, architectural and technical requirements[4].

Problem Statement

Wall Street Investment bank [4] decides to build an efficient bond pricing system. If traders want to buy bonds they will have to send prices to different trading venues[4]. Each trading center had their own user interface. This system will compare bond price with different markets and get the best price for user. In order to get best request system needs to communicate with different components. For this we need the integration of various enterprise software which makes them to process the communication between the components. for this we have used an open source integration framework which implements the various Enterprise integration patterns and thus providing the routes and mediation logic.

High Level Flow [4], System will receive market data. Then analytics engine will alter the data. Modified analytical data is sent out to different trading venues [4] so that all traders can buy or sell the bonds.

Design process problems: Java Client applications were implemented because it's more users friendly and it's compatible with Windows NT and Solaris workstations. C++ components were used on server side. TIBCO information Bus messaging infrastructure will communicate with market data components [4]. The process of communication and integration of all these components is considered to be the real design process problem.

Components in the integration process
Three major components in the integration process are

- Market Data Price Feed Server [4]
- Analytics Engine [4]
- Contribution Server [4]

The main problem and the solution we propose is the integration of the above components to communicate with each other.

Proposed Solution

First we decide how to integrate the Java thick client and two Java server components. We had four different integration to choose. They are File transfer, Shared Database, Remote Procedure Invocation and Messaging [4]. We decide to go with Messaging as it is easy to implement in Java and it simplifies this problem. Different types of pricing data can have separate channel with Messaging .The messaging is implemented using the open source integration framework in apache camel. This also takes care of the routing process along with the mediation logic. Thus camel is very helpful for us to provide integration in this scenario and provide the concrete implementation of the Enterprise integration patterns.

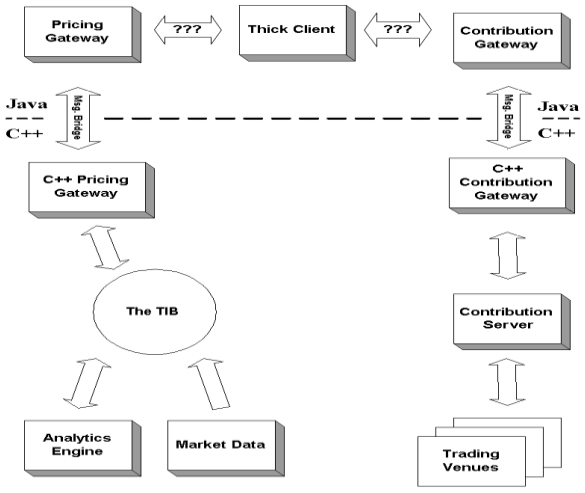


Fig 1: shows the current system design. This application pattern contains Gateways and components used in system. Channel Adapters and non-messaging protocol was used to implement Message translator. Two messaging systems are connected using Channel Adapters.[4]

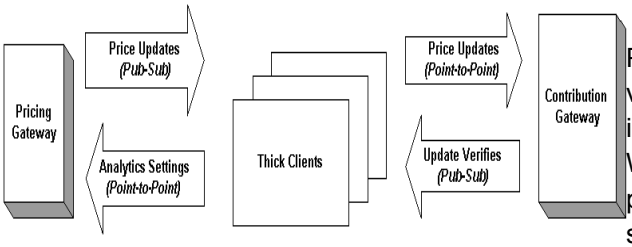


Fig 2: Message flow with Channel Types[4]

Above figure shows the Message flow with Channel Types. Message channel is used only in one direction. Point-to-Point is used for Client-to-Server communication. Publish-subscribe are used in the case of server-to-client communications. Direct communication and multicast increases the system productions.

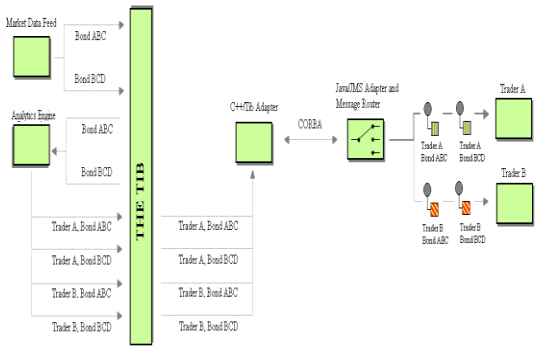


Fig 3: Market Data Flow to Client[4]

***Major Production Crash*:** System goes down due to heavy flow and pattern is used to fix problem. Message Dispatcher improved but did not fix the problem completely.[4]

Case Study Summary

Patterns are applied to solve problems in various stages. We had problem during initial design and we used pattern to solve it. We also noticed patterns are used in all third party products. Most real problems have same kind of business, technical and architectural problems and using patterns can solve it. So understanding how patterns works is very important. Once we understand how patterns works we can implement in our system to solve problems. But the implementation concern in our case is taken over by the Apache camel which makes the Enterprise Integration process simple. Thus we were able to integrate the components without any concern to the implementation of the Enterprise integration patterns because it is taken care by the open source framework call Apache camel [4]

Conclusion

Thus, one can make an Integration of Enterprise Integration Applications Possible with the help of open source integration framework called Apache camel which provides the routing methods, mediation logic and implementation of the Enterprise Integration Patterns Possible. As discussed in the paper we provide the solution to the integration of Immigration systems of various countries at large using this open source framework which will be considered flexible on the part of the developers ar they need not worry about the routing ,mediation logic and implementation of enterprise integration patterns. As a part of the research report we are providing the coding for transferring files through an ftp server using apache camel which is considered to be the implementation part of the big problem domain.

References

1. Umapathy, K.; Purao, S.; , "Designing Enterprise Solutions with Web Services and Integration Patterns," Services Computing, 2006. SCC '06. IEEE International Conference on , vol., no., pp.111-118, 18-22 Sept. 2006doi: 10.1109/SCC.2006.42. Web. 05 May 2011.<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4026911&isnumber=4026874>

2. Xinhuai Tang; Xiangfeng Luo; Xueqiang Mi; Xiaozhou Yuan; Delai Chen; , "DSL Route: An Efficient Integration Solution for Message Routing," Semantics, Knowledge and Grid, 2009. SKG 2009. Fifth International Conference on , vol., no., pp.436-437, 12-14 Oct. 2009

doi: 10.1109/SKG.2009.66. Web. 05 May 2011.<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5370324&isnumber=5368022>

3. "Apache Camel: User Stories." Apache Camel: Index. The Apache Software Foundation. Web. 05 May 2011. <http://camel.apache.org/user-stories.html>.

4. Simon, Jonathan. "Enterprise Integration Patterns - Case Study: Bond Trading System." Home - Enterprise Integration Patterns. 2003. Web. 09 May 2011. <http://www.eaipatterns.com/BondTradingCaseStudy.html>.

5. Anstey, Jonathan. Apache Camel: Integration Nirvana. Progress Software Corporation, 20 Mar. 2009. Web. 5 May 2011. <http://architects.dzone.com/sites/all/files/DZone_Camel_Article_JonathanAnstey.pdf>.

6. "Introduction to Apache Camel." Interview by Ibsen Claus. Fusesource.com. Progress Fuse, Sept. 2010. Web. 5 May 2011. <http://download.progress.com/5331/open/adobe/prc/products/fuse/intro_to_apache_camel/index.htm>.

7. Reijn, Jeroen. "Apache Camel: Open Source Integration Framework." Web log post. Jeroen Reijn. 30 Mar. 2009. Web. 05 May 2011. <http://blog.jeroenreijn.com/2009/03/apache-camel-open-source-integration.html>.

8. Suster, Mark. "Social Networking." Past, Present, Future (2010). GRP Partners, 9 Oct. 2010. Web. 05 May 2011. <8. https://docs.google.com/viewer?a=v&pid=gmail&attid=0.1&thid=12fabef24bd35692&mt=application/vnd.ms-powerpoint&url=https://mail.google.com/mail/?ui%3D2%26ik%3D7e0577c544%26view%3Datt%26th%3D12fabef24bd35692%26attid%3D0.1%26disp%3Dattd%26realattid%3Df_gn62gvc00%26zw&sig=AHIEtbT5LUl_SD2f-Y9Oi3yPoDo4YIYdcA>.

9. Apache Camel: Index. Web. 09 May 2011. <http://camel.apache.org/>