**CHAPTER 1**

**INTRODUCTION**

**1.1 OVERVIEW**

EFM Operational and Business Intelligence (OBI) is a set of near real-time Financial Messaging Data Marts (DM) for EFM’s (Enterprise Financial Messaging) systems. EFM’s primary reason is messaging. The database is a repository for information related to financial messages that flow through EFM.

A data mart is the access layer of the data warehouse environment that is used to get data out to the users. The data mart is a subset of the data warehouse that is usually oriented to a specific business line or team.

Data polled in from different systems to do analytics and business transactions is called Data Warehousing.

**1.2 OBJECTIVE OF THE PROJECT**

The main objective of this project is to understand XML Message Parsing and Ways to Load it into Database using Hibernate.

Other objectives are while reading data and writing to database the speed should be greater than forty messages per second from the queue. Multithreading is implemented so that concurrency can be achieved. XML should write to an in-memory cache so that time can be saved. This is because reading and writing from memory is faster than reading and writing from disc. The system should not fail in case of changes in the database table. The overall objective of this project is to reduce the expenses for parsing and writing messages.

**1.3 NEED FOR THE STUDY**

As the traditional message loading is slow and tedious a new method to automatically load the messages into the database is needed. Also the time to read messages should be improved from 20 messages per second to 40 messages per second to enhance the efficiency.

**1.4 SCOPE OF THE PROJECT**

This project has a high scope in Enterprise Financial Messaging domain. In this domain many message transactions are to be carried out every day and hence performance plays an important role. A system that is fast and has no down time is needed. Thus in this project we include multi-threading concept so that if there is a volume spike the system doesn’t become slow.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 TITLE:** Integration of Struts, Spring and Hibernate for an University Management System

**AUTHORS**: Ankur Bawiskar , Prashant Sawant , Vinayak Kankate , Dr. B.B.Meshram

This paper presents the concept of the Spring framework which is widely used in making enterprise wide applications along with other frameworks. Spring framework insists that ordinary java beans can be used with slight modifications to them. This framework is used within J2EE architecture to make it easier to develop large scale java applications. This paper presents the architecture overview of spring framework along with the features of the framework that have made the framework useful for using it along with struts and hibernate. The integration of various frameworks for an E-commerce system such as university system has also been discussed in the paper. The Spring MVC framework as an important part is also discussed along with struts mvc. This paper has a proposed architecture for a website based system using the integration of Spring, Hibernate and Struts framework.The Spring framework described above which has a multitier architecture can be effectively used with many other frameworks such as Struts which can be used for web presentation layer and can be easily integrated with Hibernate persistence framework which is a powerful database connectivity layer. Although spring has its own MVC layer it can easily interoperate with the presentation tier of Struts i.e. Struts mvc. The proposed system here can be considered here with the help of a website such as that of a University management system wherein students can enroll for a course and can also avail the e-learning facility provided by the university. Students can also download papers, research papers. It can generally have three layers i.e. presentation layer, business layer and database layer.

**2.2 TITLE:** XML Processing in J2EE

**AUTHORS**: Biswadeep Nag

In the modern world of web service enabled enterprise applications,the ability to parse and process XML documents efficiently is of fundamental importance. J2EE offer extensive capabilities for processing XML. In this paper, we make certain technical comparisons about the different XML processing techniques that are offered. We discuss the basics of the different parsing alternatives available in the two platforms such as SAX, DOM, pull parsing, as well as the XML binding technologies JAXB and XmlSerializer. We then utilize a suite of open-source XML processing programs that explores various components of the parsing infrastructure and offer examples of which technique might be most suitable during the lifecycle of an XML document.

**2.3 TITLE:** Java facilities in processing XML files – JAXB

**AUTHORS**: Dănuţ-Octavian SIMION

The paper presents the Java programming language facilities in working with XML files using JAXB (The Java Architecture for XML Binding) technology. The XML file can be an existing one and could contain the data about an entity (Clients for example) or it might be the result of a SELECTSQL statement. JAXB generates JAVA classes through xs rules and a Marshalling, Unmarshalling compiler. Java facilities in processing XML files – JAXB (The Java Architecture for XML Binding) is used for extracting data from an existing XML file into Java classes and from this point, the objects can be integrated in different types of applications . The principals facilities of JAXB are: Hides the necessary details of processing data. Offers a modality object oriented in working with XML files. Generating the Java classes is based on xs rules through a Marshalling, Unmarshalling compiler. Allows separating activities for programmers and web designers. Here is an of a XML file that contain data about the entity Clients.

**2.4 TITLE:** Document Transformation System from Papers to XML Data

**AUTHORS:** Yasuto ISHITANI

This paper proposes a new method for document transformation using OCR to generate various XML documents from printed documents. The proposed method adopts a hierarchical transformation strategy based on a pivot XML document. Firstly, document elements such as title, authors, abstract, headings, paragraphs, lists, captions, tables and figures are extracted from document images. Secondly, the hierarchical structure of document elements is extracted and is described using a DOM tree. Thirdly, this document structure is converted into a pivot XML document described as an XHTML document by an XML parser. Finally, this pivot XML document is transformed into the target XML document by the XML parser with XSLT scripts or specific programs. Experimental results show the method is effective in transforming printed documents to various XML documents. A document transformation system for transforming printed documents to various XML documents was proposed. The proposed system adopted a hierarchical document transformation strategy based on a pivot XML document. In this strategy, a target XML document was hierarchically obtained from the pivot XML document that was transformed from the hierarchical structure of document elements in the printed documents. The pivot XML document was based on the XHTML DTD and had powerful expressivities sufficient to represent the contents of the various printed documents. According to this hierarchical strategy, it was possible to obtain several different types of XML document from the same pivot XML document. As a result, this system allows printed documents to be used in the field of e-government, e-books/e-publishing, digital libraries, and knowledge management by utilizing XML technologies. Experiments involving a variety of documents have shown that the method is effective for various types of printed documents.

**2.5 TITLE:** Requirements for XML Document Database Systems

**AUTHORS:** Airi Salminen

Many XML documents will be transient representations for the purpose of data exchange between different types of applications, but there will also be a need for effective means to manage persistent XML data as a database. In this paper we explore requirements for an XML database management system. The purpose of the paper is not to suggest a single type of system covering all necessary features. Instead the purpose is to initiate discussion of the requirements arising from document collections, to offer a context in which to evaluate current and future solutions, and to encourage the development of proper models and systems for XML database management. Our discussion addresses issues arising from data modelling, data definition, and data manipulation. The purpose of the paper is to initiate discussion of the requirements for XML databases, to offer a context in which to evaluate current and future solutions, and to encourage the development of proper models and systems for XML database management. A well-defined, general-purpose XML database system cannot be implemented before database researchers and developers understand the needs of document management in addition to the needs of more traditional database applications.

**CHAPTER 3**

**SYSTEM ANALYSIS**

**3.1 PROBLEM DEFINITION**

The problem definition of this project is to illustrate the detailed requirements of “**DEVELOPING HIGH PERFORMANCE INHOUSE MESSAGE LOADERS**”. This will explain the purpose and features of the system, what the system will do, and the constraints under which it must operate.

**3.2 EXISTING SYSTEM**

A plain JAVA based parser which transforms only 10 – 15 messages per second. Traditional batch-oriented ETL processing is used.

Batchprocessing is the execution of a series of [programs](https://en.wikipedia.org/wiki/Computer_program) on a [computer](https://en.wikipedia.org/wiki/Computer) without manual intervention.  The execution of a series of programs each on a set or "batch" of inputs, rather than a single input (which would instead be a custom [job](https://en.wikipedia.org/wiki/Job_(computing))). This method can be used only when there is less traffic and less transactions are involved in the process.

Frequency Batch Processing is the transaction of a group of messages at regular intervals. There will be delays in message transactions , for example messages are trans actioned once in every 15 minutes interval.

**3.3 PROPOSED SYSTEM**

An event-based, streaming architecture has been adopted for ETL (Extract, Transform, Load) to facilitate near real-time information access. The Extract component is provided by source systems via MQ (Message Queueing). Message Queue is a reliable asynchronous messaging service that conforms to the JMS 1.1 specification. In addition, to provide for the needs of large-scale enterprise deployments, Message Queue provides a host of features that exceed JMS specification requirements.  Transform and Load processing have been separated so a single, standard Loader can be used, regardless of format of the data source.  The data streams into the Real-time (RT) database.  This is JABX based system which transforms more than 40 messages per second. Real Time Loading process is used where there is no delay in the messages being trans actioned.

**3.4 FEASIBILITY STUDY**

Feasibility study is an evaluation of system regarding to its workability , impact on organization, ability to meet user needs and effective use of resources. All projects are feasible given unlimited resources and infinite time. The feasibility study is to determine whether the solution is achievable, given the organizations resource and constraints. By performing the feasibility study the scope of the system will be defined completely. It is both necessary and prudent to evaluate the feasibility of the project at the earliest possible time. Feasibility and risk analysis is related in many ways. If project risk is great the feasibility listed below is equally important.

The following feasibility techniques has been used in this project

* Technical feasibility
* Operational feasibility
* Economic feasibility

**3.4.1Technical Feasibility**

Technical feasibility analysis makes a comparison between the levels of technology available that is needed for the development of the project. The level of technology consists of factors like software tools, machine environment, platform developed and so on. Technical feasibility centers around the hardware and software and to what extent it can be deployed to run successfully. The tools that are used to develop the application are the best tools available in the technological scenario and hence it requires efficient and versatile programmers and programming skills .Maven, Hazelcast, Hibernate, oracle database and an IDE are sufficient for the project to be technically feasible.

**3.4.2 Economic Feasibility**

Economic feasibility is used for evaluating the effectiveness of the system. The purpose of the economic feasibility assessment is to determine the positive economic benefits to the organization that the proposed system will provide. It includes quantification and identification of all the benefits expected. This assessment typically involves a cost/ benefits analysis. All the softwares involved in the project are open source.  So it can be easily downloaded from the internet. So no high cost is involved. Regarding the maintenance, since the source code will be with us , any small and necessary changes can be done with minimum maintenance cost involved in it. Hence the system is economically feasible and the risk of finance doesn’t exist.

**3.4.3 Operational Feasibility**

Operational feasibility is necessary as it ensures the success of the project. Certain tests have been carried out to ensure the operational feasibility of the system. The proposed system operates well with all types of connection. All the capabilities work well if proper functioning environment is provided. Hence the system is operationally feasible.

**3.5 System Configuration**

**3.5.1 Introduction**

The requirement specification is a technical specification of requirements for the software products. It is the first step in the requirement analysis process. It lists the requirements of a particular software system including functional, performance and security requirements. The requirements also provide usage scenarios from a user, an operational and an administrative perspective. The purpose of software requirement specification is to provide a detailed overview of software projects, its parameters and goals. This describes the project target audience and its user interface, hardware and software requirements. It defines how the client, team and audience see the project and its functionalities.

**3.5.2 Hardware and Software Requirements**

**3.5.2.1 Hardware Requirements**

Processor : Any processor above 500MHz

Ram : 128 Mb

Hard disk : 10 GB

Compact disk : 650 Mb

Input device : Standard Keyboard and Mouse

Output device : Display screen

**3.5.2.2 Software Requirements**

Operating System : Windows Family

Language : Java

Front End : Eclipse

Back End : Oracle SQL Developer

**CHAPTER 4**

**TECHNOLOGIES USED**

**4.1 JAVA**

Java is a computer programming language that is concurrent, class based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers “write once, run anywhere” (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java applications are typically compiled to bytecode (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture.

Java is a simple, object-oriented, network-savvy, interpreted, robust, secure, architecturally neutral, portable, high-performance, multithreaded, dynamic language.

**4.2 ECLIPSE**

In computer programming, Eclipse is an integrated development environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. Written mostly in Java, Eclipse can be used to develop applications.

By means of various plug-ins, Eclipse may also be used to develop applications in other programming languages: Ada, C, C++, COBOL. It can also be used to develop packages. Development environments include the Eclipse Java development tools (JDT) for Java.

**4.3 ANALYSED TECHNOLOGIES**

**4.3.1 XML**

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format which is both human-readable and machine-readable. It is defined by the W3C's XML 1.0 Specificationand by several other related specifications, all of which are free open standards.

The design goals of XML emphasize simplicity, generality and usability across the Internet. It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures such as those used in web services.

Several schema systems exist to aid in the definition of XML-based languages, while many application programming interfaces (APIs) have been developed to aid the processing of XML data.

**4.3.2 SPRING**

The Spring Framework is an application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform. Although the framework does not impose any specific programming model, it has become popular in the Java community as an alternative to, replacement for, or even addition to the Enterprise JavaBeans(EJB) model. The Spring Framework is open source.

**4.3.3 JAXB**

Java Architecture for XML Binding (JAXB) allows Java developers to map Java classes to XML representations. JAXB provides two main features: the ability to marshal Java objects into XML and the inverse, i.e. to unmarshal XML back into Java objects. In other words, JAXB allows storing and retrieving data in memory in any XML format, without the need to implement a specific set of XML loading and saving routines for the program's class structure

JAXB is particularly useful when the specification is complex and changing. In such a case, regularly changing the XML Schema definitions to keep them synchronized with the Java definitions can be time consuming and error-prone.

JAXB is a part of the Java SE platform and one of the APIs in the Java EE platform, and is part of the Java Web Services Development Pack (JWSDP). It is also one of the foundations for WSIT.

**4.3.4 HIBERNATE**

Hibernate ORM (Hibernate in short) is an object-relational mapping framework for the Java language. It provides a framework for mapping an object-oriented domain model to a relational database. Hibernate solves object-relational impedance mismatch problems by replacing direct, persistent database accesses with high-level object handling functions.

Hibernate is free software that is distributed under the GNU Lesser General Public License 2.1.

Hibernate’s primary feature is mapping from Java classes to database tables; and mapping from Java data types to SQL data types. Hibernate also provides data query and retrieval facilities. It generates SQL calls and relieves the developer from manual handling and object conversion of the result set.

**4.3.5 MAVEN**

Maven is a build automation tool used primarily for Java projects. The word maven means "accumulator of knowledge". Maven addresses two aspects of building software: First, it describes how software is built, and second, it describes its dependencies. Contrary to preceding tools like Apache Ant, it uses conventions for the build procedure, and only exceptions need to be written down. An XML file describes the software project being built, its dependencies on other external modules and components, the build order, directories, and required plug-ins. It comes with pre-defined targets for performing certain well-defined tasks such as compilation of code and its packaging.

Maven dynamically downloads Java libraries and Maven plug-ins from one or more repositories such as the Maven 2 Central Repository, and stores them in a local cache.  This local cache of downloaded artifacts can also be updated with artifacts created by local projects. Public repositories can also be updated.

The Maven project is hosted by the Apache Software Foundation.

Maven is built using a plugin-based architecture that allows it to make use of any application controllable through standard input. Theoretically, this would allow anyone to write plugins to interface with build tools (compilers, unit test tools, etc.) for any other language. In reality, support and use for languages other than Java has been minimal.

**4.3.6 HAZELCAST**

In computing, Hazelcast is an open source in-memory data grid based on Java.

In a Hazelcast grid, data is evenly distributed among the nodes of a computer cluster, allowing for horizontal scaling both in terms of available storage space and processing power. Backups are also distributed in a similar fashion to other nodes, based on configuration, thereby protecting against single node failure.

Typical use-cases for Hazelcast include:

* Distributed cache, often in front of a database
* storage for temporal data, like web sessions
* in-memory data processing and analytics
* Cross-JVM communication and shared storage

**CHAPTER 5**

**SYSTEM DEVELOPMENT**

**5.1 MODULE DESCRIPTION**

Database

Queues

Hibernate Hazelcast cache

Parser Program

**Fig 5.1 Module Description**

This provides detailed information about each of the modules and its supported components.

**5.1.1 QUEUES**

A queue is an example of a linear data structure, or more abstractly a sequential collection. A collection designed for holding elements prior to processing. Besides basic [Collection](https://docs.oracle.com/javase/7/docs/api/java/util/Collection.html) operations, queues provide additional insertion, extraction, and inspection operations. Each of these methods exists in two forms: one throws an exception if the operation fails the other returns a special value (either null or false, depending on the operation). Queues typically, but do not necessarily, order elements in a FIFO (first-in-first-out) manner.

Queues are implemented as containersadaptors, which are classes that use an encapsulated object of a specific container class as its underlyingcontainer, providing a specific set of member functions to access its elements. Elements are pushed into the *"*back*"* of the specific container and popped from its *"*front*"*.

**5.1.2 PARSER PROGRAM / JAXB PARSING**

XML Parser provides way how to access or modify data present in an XML document. Java provides multiple options to parse XML document. There are JAXB and XSLT APIs available to handle XML parsing in Object Oriented way.

The messages arrive to this system as a canonical XML. Canonical XML is a standardized version of the message and is useful for the Program since the Program has to deal with only one type of system. It doesn’t have to deal with multiple message formats. The incoming messages from external environment will be transformed into Canonical message using a Transform tool like Volante and is presented as an input to the Parser program. The parser program then takes up the Canonical XML and unmarshals it into Java Object. There are multiple Java XML Parsers available. Few of them are given below

* SAX Parser
* DOM Parser
* JAXB Parser

SAX Parser is used to marshal smaller and simpler XMLs effectively. However, it is not effective when the XMLs are complex and volume is larger.

DOM Parser utilizes a lot of memory space and is very slow when there is a huge volume of XML. JAXB Parser is the industry standard for parsing the XML messages and is faster compared to the other two XML parser.

Working with JAXB is easy, just annotate object with JAXB annotation, later use jaxbMarshaller.marshal() or jaxbMarshaller.unmarshal() to do the object / XML conversion.

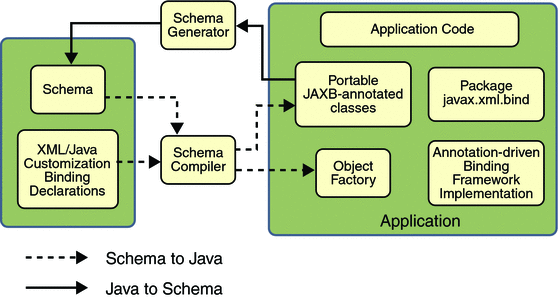
A JAXB implementation consists of the following architectural components:

Schemacompiler:

Binds a source schema to a set of schema-derived program elements. The binding is described by an XML-based binding language.

Schemagenerator:

Maps a set of existing program elements to a derived schema. The mapping is described by program annotations.



**Fig 5.1.2 Jaxb Architecture**

Bindingruntimeframework:

Provides unmarshalling (reading) and marshalling (writing) operations for accessing, manipulating, and validating XML content using either schema-derived or existing program elements.

The process has four main steps that are explained as follows:

1. JAXB Dependency

No extra jaxb libraries are required if you are using JDK1.6 or above, because JAXB is bundled in JDK 1.6.

2. JAXB Annotation

For object that needs to be converted to / from XML file, it has to be annotated with JAXB annotation. The most convenient way to obtain the Java type information describing the node elements is by compiling an XML schema, usually written in the [W3C XML Schema Language](http://www.w3.org/TR/xmlschema-0), using the JAXB Binding Compiler xjc. The resulting set of classes defines the types required for accessing elements, attributes and other content in a type safe way.

3. Convert Object to XML

JAXB marshalling example, convert customer object into a XML file. The jaxbMarshaller.marshal() contains a lot of overloaded methods, find one that suit your output.

File file = new File (“C:\\fle.xml”);

JAXBContext jaxbContext = JAXBContext.newInstance (Customer.class);

Marshaller jaxbMarshaller = jaxbContext.createMarshaller();

jaxbMarshaller.marshal(customer, file);

jaxbMarshaller.marshal(customer, Sytem.out);

## 4. Convert XML to Object

JAXB unmarshalling example, convert a XML file content into a customer object. The jaxbMarshaller.unmarshal()contains a lot of overloaded methods, find one that suit yours.

File file = new File(“C:\\file.xml”);

JAXBContext jaxbContext = JAXBContext.newInstance(Customer.class);

Unmarshaller jaxbUnmarshaller = jaxbContext.createUnmarshaller();

Customer customer = (Customer) jaxbUnmarshaller.unmarshal(file);

**5.1.3 HIBERNATE HAZELCAST CACHE**

Hibernate is a Java ORM (Object Relational Model) to convert Java Objects into Relational Database model. This is done by creating Getter/Setter methods of the objects on the relational model. The Java Object is read and transformed into the Relational Objects by setting the values in the Relational objects from the Java Objects created by unmarshalling the XML. The resultant Relational Object is then loaded into the database using Hibernate itself.

## Create POJO Classes:

The first step in creating an application is to build the Java POJO class or classes, depending on the application that will be persisted to the database.

A POJO (Plain Old Java Object) is a Java object that doesn't extend or implement some specialized classes and interfaces respectively required by the EJB framework. All normal Java objects are POJO.

public int getId()

{

return id;

}

public void setId(int id)

{

this.id = id;

}

## Create Database Tables:

Second step would be creating tables in your database. There would be one table corresponding to each object you are willing to provide persistence.

## Create Mapping Configuration File:

This step is to create a mapping file that instructs Hibernate how to map the defined class or classes to the database tables. You should save the mapping document in a file with the format <classname>.hbm.xml.

The mapping document is an XML document having <hibernate-mapping> as the root element which contains all the <class> elements.

The **<class>** elements are used to define specific mappings from a Java classes to the database tables.

The **<id>** element maps the unique ID attribute in class to the primary key of the database table. The **name** attribute of the id element refers to the property in the class and the **column** attribute refers to the column in the database table. The **type** attribute holds the hibernate mapping type, this mapping types will convert from Java to SQL data type.

The **<generator>** element within the id element is used to automatically generate the primary key values. Set the **class** attribute of the generator element is set to **native** to let hibernate pick up identity**, sequence** or **hilo** algorithm to create primary key depending upon the capabilities of the underlying database.

The **<property>** element is used to map a Java class property to a column in the database table. The **name** attribute of the element refers to the property in the class and the **column** attribute refers to the column in the database table. The **type** attribute holds the hibernate mapping type, this mapping types will convert from Java to SQL data type.

## Create Application Class:

Finally, we will create our application class with the main() method to run the application.

factory = new Configuration().configure().buildSessionFactory();

## Compilation and Execution:

Here are the steps to compile and run the above mentioned application. Make sure you have set PATH and CLASSPATH appropriately before proceeding for the compilation and execution.

* Create hibernate.cfg.xml configuration file as explained in configuration chapter.
* Create Employee.hbm.xml mapping file as shown above.
* Create Employee.java source file as shown above and compile it.
* Create ManageEmployee.java source file as shown above and compile it.
* Execute ManageEmployee binary to run the program.

**5.1.4 DATABASE**

The Database schematic follows Star Schema. Star Schema is a type of Datawarehouse modelling where one fact is surrounded by multiple dimensions.

The incoming XML has to be loaded to both Dimension table and the Keys of the dimension table needs to be carried to Fact Table.

Date Dimension

Application Dimension

Message Fact Table

Message Format

**Fig 5.1.4 Database Model**

Oracle Database

Queues

Hibernate HazelCast

JAXB Parser Instances

**Fig 5.1.5 Technical Loading Process**

**CHAPTER 6**

**SYSTEM DESIGN**

System design is the process or art of defining the hardware and software architecture, components , modules, interfaces and data for a computer system to satisfy specified requirements. One could see it as an application of system  theories to computing. Some overlap with the discipline of system analysis appears inevitable.Design tools such as Unified Modelling Language (UML), now addresses   some of the issues of computer systems design and interfacing.

**6.1 ARCHITECTURE DIAGRAM**

Queues – Queues are given as input for financial messaging. The queues can be either in SWIFT (Society for Worldwide Interbank Financial Telecommunication) or FED (Federal Department of United States) or CHIPS (Clearing House Interbank Payment System) format. CHIPS transactions are dealing with domestic US (United States) transactions. FED is used for high value payment transactions. SWIFT is used for low value transactions. We consider only SWIFT messages and hence they are given in the form of queue as input.

ReadMessage() – This class will read the messages from the Queue and will call the ParseMessage object with the canonical XML file as an input.

ParseMessage() - This object will operate upon the XML file input and convert it into Java Objects. We use jaxb parsing to convert the XML input into java objects.

WriteToMemory() - These Java Objects will be passed to writetoMemory class which would then store this message in hazelcast. In-memory cache is used in hazelcast for efficient storing.

PersistMessage() - This will read from the memory and store it in the database. Once the data is stored in database it removes the record from the memory.

ReadMessages()

Queues

PersistMessage()

WriteToMemory()

ParseMessages()

**Fig 6.1 Detailed Architecture**

**6.2 GENERAL ARCHITECTURE**

OBI

CHIPS

FED

SWIFT

gateway

EFM

OFAC

Transformation

**Fig 6.2 Process Flow**

EFM is used by the banks across the world as a means of communication. Banks that are connected together and have a direct communicational link form the Financial Network. Two banks having no direct communication between them uses a correspondent bank which acts as a server to establish the relation .The common language used between the banks is known as Financial Messaging. Examples are SWIFT, FED, CHIPS

The SWIFT messages are organized as a queue and fed into the gateway of the EFM. The input should be in the standard XML format known as canonical XML(CXML).Every transaction has to be validated by the OFAC (Office Of Foreign Assets Control).The validation is done using DNBW (Do Not do Business With) lists. Hence checks are imposed to see if the transactions being made are legal. Once the validation is over the XMLs are converted into java objects in the transformation phase and then the data is loaded into the database.

The OBI (Operational and Business Intelligence) is a data ware house that generates the reports of the overall transactions being made. It is an OLAP (Online Analytical Processing) System. **OLAP** performs multidimensional analysis of business data and provides the capability for complex calculations, trend analysis, and sophisticated data modeling. It answers multidimensional analytical queries swiftly.

**6.3 UML DIAGRAMS**

The Unified Modeling Language (UML) is a standard visual modeling language intended to be used formodeling business and similar processes, analysis, design, and implementation of software-based systems**.** UML is a common language for business analysts, software architects and developers used to describe, specify, design, and document existing or new business processes, structure and behavior of artifacts of software systems**.**

The UML diagrams used in this project are:

* + Activity Diagram
  + Collaboration Diagram

**6.3.1** **ACTIVITY DIAGRAM**

Activity diagram is used to describe the dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements like fork, join etc.

Parse Messages

Read Messages

Queues

XML Input

Write To Memory

Persist Message

**Fig 6.3.1 Activity Diagram**

**6.3.2 COLLABORATION DIAGRAM**

A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles. The messages between objects are shown as arrows connecting the relevant rectangles along with labels that define the message sequencing.

Queues

Jaxb Parser Instances

2: Write Threads

1: Read Threads

Oracle Database

**Fig 6.3.2 Collaboration Diagram**

**CHAPTER 7**

**IMPLEMENTATION AND MAINTENANCE**

**7.1 IMPLEMENATION**

This project is implemented in organization to improve the performance of reading and fetching the messages. It provides a way for organizing the messages in a systematic way.

**7.2** **MAINTENANCE**

System Maintenance is that phase of software engineering that helps the user and the developer to maintain the software in an environment required by the user. It is this phase in which any sort of implementation notification are made to incorporate the changes that the user need.

**CHAPTER 8**

**TESTING**

**8.1 INTRODUCTION**

After finishing the development of any computer based system the next complicated time consuming process is system testing. During the time of testing only the development company can know that, how far the user requirements have been met out, and so on.

Following are the some of the testing methods applied to this effective project:

**8.2 CODE GENERATION TEST**

The code-generation test exercises two main aspects of the code generation, The XML Schema feature support. Any tool may have weaknesses in that features may be unsupported, or poorly supported. Flaws in the generated Java Code. Once the code has been generated, it is compiled in a java compiler. Code must compile to be usable.

**8.3 RUN TIME TEST**

The runtime tests consists of a "roundtrip", that is, converting the XML document to populated Java objects, converting those Java objects back to an XML instance. It also includes comparing the input and output. This is a good test for two important aspects of the ultimate functionality of the generated code. It should be noted that this comparison only compares the basic functionality, without any customization of the generated code. In spite of the fact that the greatest parameter has XGen tool, in the offered approach has been chosen JAXB since his features as much as possible correspond to requirements in realization of specialized schemes.

# 8.4 SOURCE CODE TESTING:

This examines the logic of the system. If we are getting the output that is required by the user, then we can say that the logic is perfect.

**8.5 SPECIFICATION TESTING:**

We can set with, what program should do and how it should perform under various condition. This testing is a comparative study of evolution of system performance and system requirements.

**8.6 MODULE LEVEL TESTING:**

In this the error will be found at each individual module, it encourages the programmer to find and rectify the errors without affecting the other modules.

**8.7 UNIT TESTING:**

Unit testing focuses on verifying the effort on the smallest unit of software-module. The local data structure is examined to ensure that the date stored temporarily maintains its integrity during all steps in the algorithm’s execution. Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing.

**8.8 INTEGRATION TESTING:**

Data can be tested across an interface. One module can have an inadvertent, adverse effect on the other. **Integration testing** is a systematic technique for constructing a program structure while conducting tests to uncover errors associated with interring.

**8.9 VALIDATION TESTING:**

It begins after the integration testing is successfully assembled. Validation succeeds when the software functions in a manner that can be reasonably accepted by the client. In this the majority of the validation is done during the data entry operation where there is a maximum possibility of entering wrong data. Other validation will be performed in all process where correct details and data should be entered to get the required results.

**8.10 RECOVERY TESTING:**

**Recovery Testing** is a system that forces the software to fail in variety of ways and verifies that the recovery is properly performed. If recovery is automatic, re-initialization, and data recovery are each evaluated for correctness.

**8.11 SECURITY TESTING:**

Security testing attempts to verify that protection mechanism built into system will in fact protect it from improper penetration. The tester may attempt to acquire password through external clerical means, may attack the system with custom software design to break down any defences to others, and may purposely cause errors.

**8.12 PERFORMANCE TESTING:**

Performance Testing is used to test runtime performance of software within the context of an integrated system. Performance test are often coupled with stress testing and require both software instrumentation.

**8.13 BLACKBOX TESTING:**

**Black-box testing** focuses on functional requirement of software. It enables to derive est. of input conditions that will fully exercise all functional requirements for a program.

Black box testing attempts to find error in the following category:

* Incorrect or missing function
* Interface errors
* Errors in data structures or external database access and performance errors.

**8.14 OUTPUT TESTING:**

After performing the validation testing, the next step is output testing of the proposed system since no system would be termed as useful until it does produce the required output in the specified format. **Output format** is considered in two ways, the **screen format** and the **printer format.**

**8.15 USER ACCEPTANCE TESTING:**

User Acceptance Testing is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system users at the time of developing and making changes whenever required.

**CHAPTER 9**

**PERFORMANCE MEASURES**

The number of messages processed over a given interval of time is called system throughput. A business enterprise must identify its projected peak message rate in order to assess whether the z/TPF system is an appropriate solution or not.The instructions to be executed over the peak period are generally given in million instructions per second (MIPS).

Highavailability refers to a system or component that is continuously operational for a desirably long length of time. Availability can be measured relative to "100% operational" or "never failing.

There are two industrial standards:

i) 99.90% usually called the ‘3 Nines’

ii) 99.999% usually called as ‘5 Nines’

This project attains 99.999% standard which is the best industrial standard available. Without transformation one message throughput is 168 milliseconds across the data center and within the same data center it is 15 milliseconds. With transformation throughput is 500 messages per second.

Multi-threading concept is used to increase the efficiency. Whenever there is a spike in traffic the thread count can be increased. One thread can process 500 messages per second then if the thread is increased by a count of three summing up to four threads the traffic spike can be handled efficiently. Since four threads can process 2000 messages per second.

**CHAPTER 10**

**CONCLUSION**

In this project, we have enhanced the performance of the existing Financial Messaging system by introducing the concept of real time loading and by using hazelcast for automatic loading into the database. Thus the performance will be increased and system will be faster and efficient. There will be no down time for the server and thus establishing a reliable and secure transaction for the financial network in the banking domain.

**REFERENCES**

[1] http://java.sun.com/xml/jaxb/. 

[2] Hibernate is a high performance object/relational persistence and query service for Java. You can find Hibernate at <http://www.hibernate.org>

[3] Wait, B. Using XML in Oracle database applications, Nov. 1999. Oracle Corporation. http://technet.oracle.com/tech/ ml/info/htdocs/otnwp/about\_xml.htm.

[4] Elliotte Rusty Harold, Processing XML with Java, 2002

[5] Clark, J., and DeRose, S. (eds.). XML Path Language (XPath) Version 1.0, W3C Recommendation 16 November 1999. http://www.w3.org/TR/1999/REC-xpath-19991116

[6] Boyer, J. Canonical XML Version 1.0, W3C Recommendation, 15 March 2001. http://www.w3.org/TR/2001/RECxml-c14n-20010315

[7] Bray, T., Hollander, D., and Layman, A. (eds.). Namespaces in XML, World Wide Web Consortium, 14 January 1999. http://www.w3.org/TR/1999/REC-xml-names-19990114/.

[8] Bray, T., Paoli, J., Sperberg-McQueen, C. M., and Maler, E. (eds.). Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation 6 October 2000. http://www.w3.org/TR/2000/REC-xml-20001006/.