Net-WMS

SPECIFIC TARGETED RESEARCH OR INNOVATION PROJECT

Networked Businesses

D.8.1 – Networked architecture – J2EE compliant

(Version 1)

Due date of deliverable: June 30th, 2007

Actual submission date: July 5th, 2007

Start date of project: September 1st, 2006 Duration: 36 months

Organisation name of lead contractor for this deliverable: Wide Scope (WID – partner 10)

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)

COVER AND CONTROL PAGE OF DOCUMENT	
Project Acronym:	**Net-WMS**
Project Full Name:	Towards integrating Virtual Reality and optimisation techniques in a new generation of **Net**worked businesses in **W**arehouse **M**anagement **S**ystems under constraints
Document id:	Net-WMS D8.1 deliverable report
Document name:	D8.1 - Networked architecture – J2EE compliant
Document type (PU, INT, RE, CO)	PU
Version:	1
Submission date:	July 5th, 2007
Authors: **Organisation:** **Email:**	Filipe Carvalho Wide Scope filipe.carvalho@widescope.pt

Document type PU = public, INT = internal, RE = restricted, CO = confidential

ABSTRACT:

This document describes the J2EE distributed architecture.
It begins with an overview of the main features that a distributed network implies.
Then a schematic design of the architectural layers is presented, followed by its detailed description.

KEYWORD LIST:
Service Oriented Architecture, Java 2 Enterprise Edition, Object Relational Mapper, Data Access Object, Data Transfer Object, Java Naming and Directory Interface, Structured Query Language, eXtensible Markup Language, Enterprise Java Beans, Plain Old Java Objects, Model-View-Controller, Business-logic Object, Lightweight Directory Access Protocol, Java Server Page, Cascading Style Sheet, Hyper Text Markup Language, Asynchronous Javascript And XML, JBoss

MODIFICATION CONTROL			
Version	**Date**	**Status**	**Author**
1	05-07-2007	V1	Filipe Carvalho

Deliverable manager

- Filipe Carvalho, WID

List of Contributors

- Filipe Carvalho, WID
- Abder Aggoun, KLS OPTIM

List of Evaluators

- Adrien Lauer, PSA
- Philippe Rohou, ERCIM

TABLE of CONTENTS

Introduction

This document describes the main components and technologies of today's J2EE architecture. To motivate the design proposed in this document, KLS OPTIM and Wide Scope decided to investigate different internet technologies related to Web 1.0 and Web 2.0. A prototype of a web-based container loading and unloading was developed. The study investigated various internet technologies allowing web-based graphics visualization : (1) Web 1.0 and CCS, (2) Web 2.0 (XML, Java Script, DOM, CSS) and (3) Web 2.0 plus Scalar Vector Graphics (SVG). The results will be presented in a separate report. The aim of this study is to investigate :

- visualization of optimisation results in a Web browser for the Net-WMS project;
- interactivity and real time of user requests.

Indeed, the focus was put on the limits of the visualization technologies and on the size of flows (visualization of a new page, size of data sent by the server to the client).
The prototype developed is composed of optimisation and visualization components. The visualization was restricted to 2D visualization.

Web 2.0 (2) was recommended to be used as the main platform for the development in the Net-WMS project.

1 - Distributed architecture features

The J2EE system architecture is built on top of basic features that any large business networked environment must consider.
Furthermore it is built from the ground up to be a **service oriented architecture** (**SOA**), i.e., it provides most of its business logic encapsulated as middleware services thus enabling third-party applications to be integrated.

The networking environment features and considerations are:

- **Remote method invocations**: logic that connects a client and server via a network connection. This includes dispatching method requests, brokering of parameters, and more.

- **Load balancing**: clients must be directed to the server with the lightest load. If a server is overloaded, a different server should be chosen.

- **Transparent fail-over**: if a server crashes, or if the network crashes, clients must be rerouted to other servers without interruption of service. This shall happen fast enough according to the business criticity.

- **Back-end integration**: code needs to be written to persist business data into a database as well as integrate with legacy systems that already exist.

- **Transactions**: transactions are required to protect data integrity and consistency namely when, e.g., two clients access the same row of the database.

- **Clustering**: if the server that contains state crashes, clustering enables such state replication across all servers so that clients can use a different server.

- **Dynamic redeployment**: software upgrades can be allowed while the system is running. However, this feature may be disabled for performance reasons.

- **Clean shutdown**: if a server is required to be shut down it must happen smoothly such that clients already posting requests to the server are not abruptly interrupted.

- **Logging and auditing**: if something goes wrong a log is available for consultation in order to determine the cause of the problem, and help in its debug.

- **Threading**: having many clients connecting to a server requires the capability of processing multiple requests simultaneously. This means that the server application must be coded to be multi-threaded.

- **Object life cycle**: the objects that live within the server need to be created or destroyed when client traffic increases or decreases, respectively.

- **Resource pooling**: if a client is not currently using a server, that server's precious resources can be returned to a *pool* to be reused when other clients connect. This includes sockets (such as database connections) as well as objects that live within the server.

- **Security**: the servers and databases need to be shielded from saboteurs. Known users must be allowed to perform only operations that they have rights to perform.

- **Caching**: many objects are used by the same clients over and over again. There is no reason why they should be instantiated and destroyed on every client request. There are performance increases when such objects are cached and reused in the same context.

It is worth noting that the J2EE architecture adopted is strictly based on industry standards and best-practices. This assures an high quality environment that any developer can understand and to which can quickly contribute.

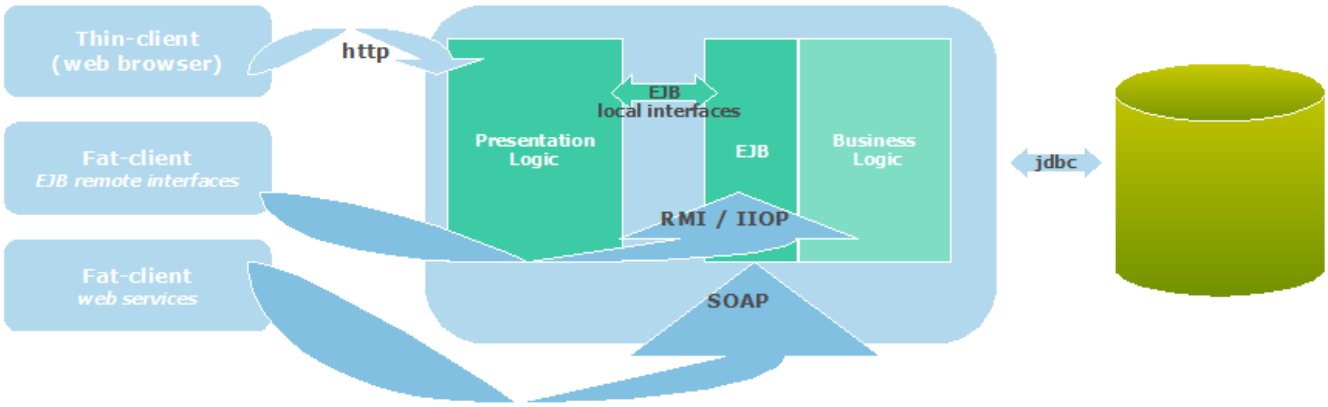
2 - Service Oriented Architecture

In Net-WMS' Service Oriented Architecture (SOA) all business logic is wrapped by a layer of EJB providing local and remote interfaces as well as a layer of web services.

All Net-WMS components are part of the business logic and so a networked environment arises from publishing such knowledge as services available for other systems to integrate with.

A set of J2EE technologies is required for implementing this concept, such as Enterprise Java Beans (EJB). For supplying the web services a framework from the Apache project is proposed. It is the Apache Axis based on SOAP for providing web services.

The following figure ilustrates the concept of a SOA applied to the Net-WMS architecture.



- Fig. 1 -

The database layer on the right is accessed through JDBC by middleware components that compose the business logic. This is where all Net-WMS solvers are integrated.

A layer of Session EJB wraps the business logic and constitute the most important integration point with external applications. It publishes the business logic as services supplied either from RMI/IIOP calls from other EJBs, and as web services to be consumed by SOAP messages.
Its local interfaces also supply content to the presentation logic layer of Net-WMS front-end.

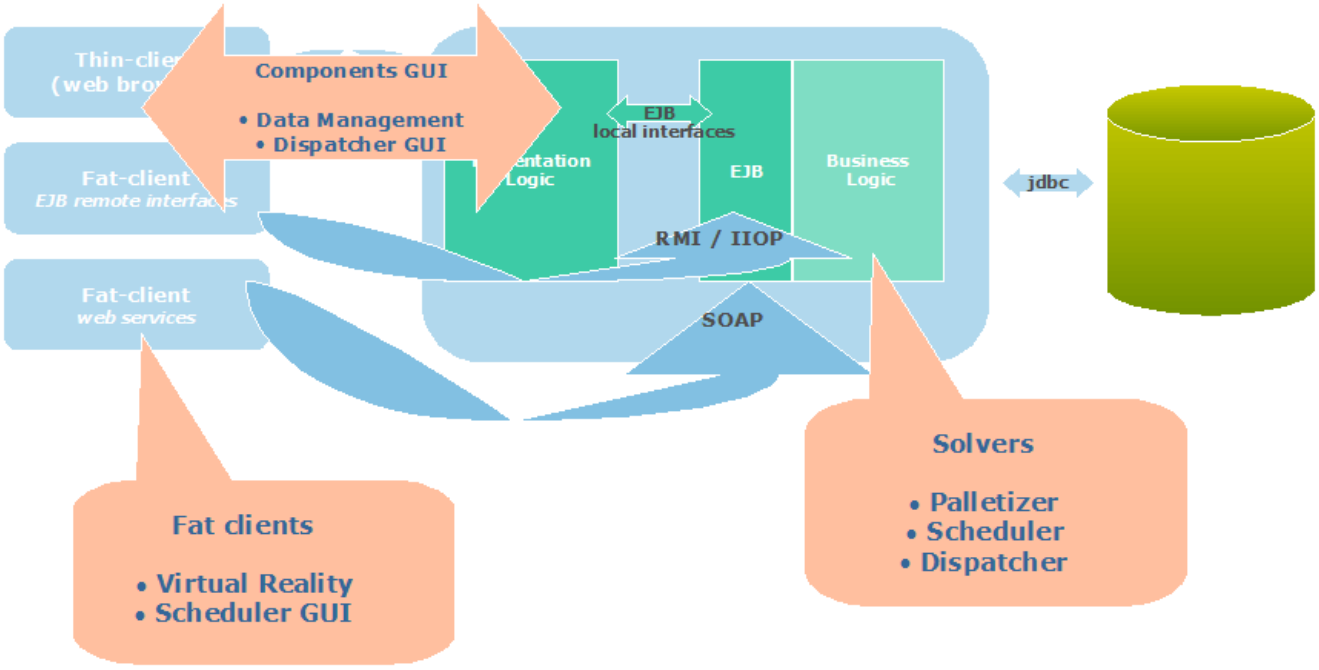
On the left several client types are depicted. A thin-client such as a user with a web browser accesses the middleware layer through HTTP. This is a typical access for a front-end user.

A fat-client, such as a standalone SWING application or even an Applet, accesses the middleware using the EJB remote interfaces through RMI / IIOP. Fat-clients may be required for user interactions that require heavy graphical computation such as the Virtual Reality viewers.

Any third-party system can also access the knowledge wrapped in the middleware through the consumption of web services based on the SOAP protocol.

3 - Net-WMS framing

The Net-WMS components fit into the proposed architecture as illustrated in the following picture.



- Fig. 2 -

The front-end data management and the dispatcher GUI are thin-client interfaces and require nothing but a web browser to run.

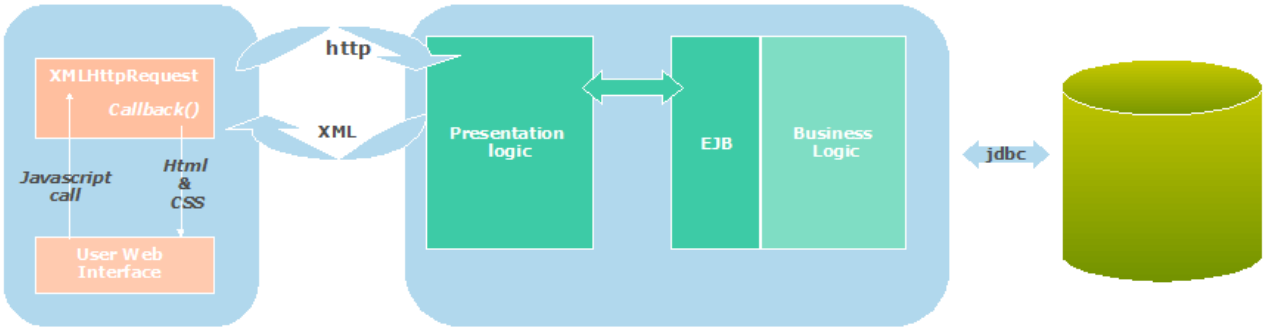
The Scheduler and the VR GUI require heavyweight clients and so they access the middleware through web services or through EJB remote interfaces.

The solvers are wrapped in the business logic layer in order to be dispatched as a set of services in the architecture.

4 – Web 2.0 and AJAX

Asynchronous JavaScript and XML (AJAX) is a technique for making the user interfaces of web applications more responsive and interactive. It is a technique that incorporates the Web 2.0 concept of providing the web user with a GUI that makes him feel like he is working with a desktop standalone application.

It increases speed and usability while providing much better user experience.
The following picture shows the scope of integrating the AJAX procedures in the Net-WMS architecture.

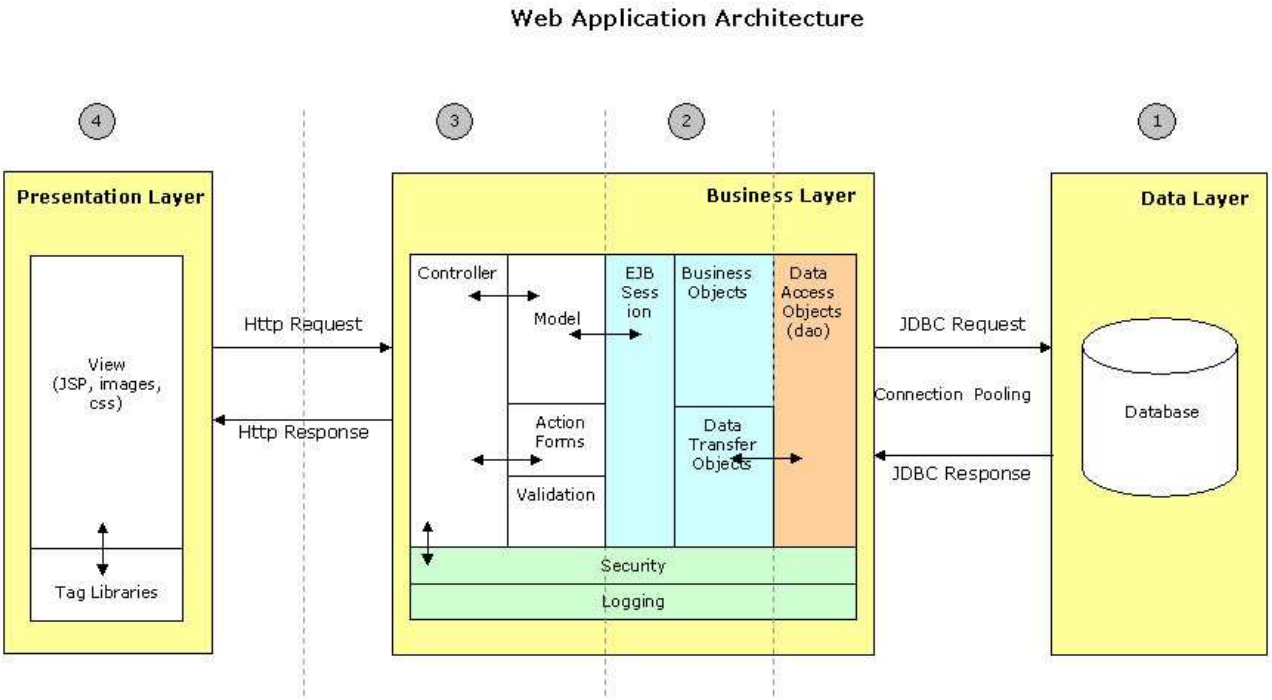


- Fig. 3 -

From the user web interface (typically an HTML page) a request is made up to the server. Such request (which may be asynchronous with any other request already being processed) is stubbed into a javascript XMLHttpRequest. This javascript object is then responsible for invoking through HTTP a server request and render an XML response.

5 – Architecture description

The web application architecture described proposes a 3-tiered approach based on Enterprise Java Beans and the Model-View-Controller paradigm.



- Fig. 4 -

It is conceived for scalability, portability, clustering and team development. In fact, it shows some layers of separation where developers can quickly progress independently of each other in a vertical development approach. Naturally, it also allows horizontal development.

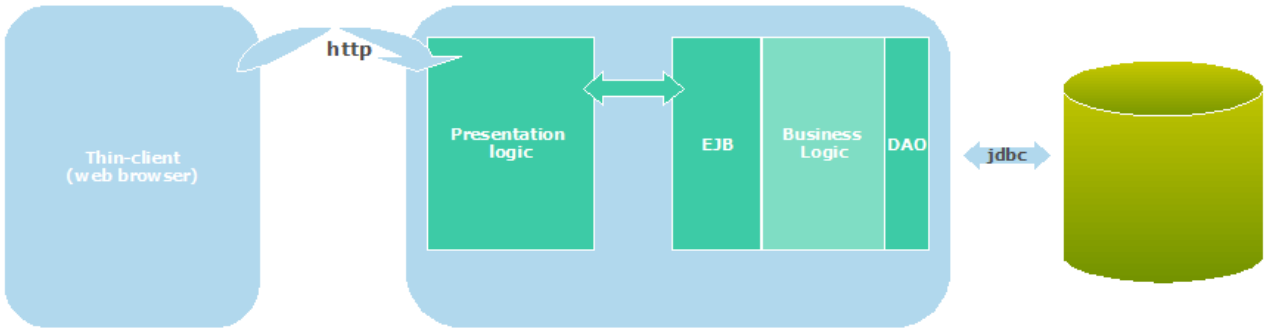
It is conceived to be compatible with any J2EE Container but specifically for JBoss. Minor changes are expected if in the future the application container must be different for some reason. Internationalization is considered from the beginning since all strings and locale-specific features are encapsulated in resource bundles.

The architecture follows a set of published and well-known J2EE design-patterns that are not fully described.

5.1 – Database Connectivity

This is a clearly separated layer of development. A user with only SQL knowledge can develop all requisites writing all SQL statements in XML files for and OR/Mapper framework to use. Additionally, a layer of Java classes (DAO) is provided, encapsulating all logic of DB access.

The following picture ilustrates where is the DAO layer positioned in the Net-WMS architecture.



- Fig. 5 -

- **Database**: Several databases are supported (PostgreSql, MySql, Oracle). The development considers PostgreSql.

- **DAO (Data Access Objects)**: Objects that encapsulate all interactions with the database. They are built on top of an Object-Relational Mapper (OR/Mapper) that transforms result sets into Java structures. The open-source framework iBatis (http://ibatis.apache.org) is integrated in order to simplify the development. In this framework all SQL statements are stored in XML files. It provides clean organization, dynamic queries and the power of SQL development.

Connections with the database are pooled and managed by the application server JBoss using an appropriate data source. This is very important, since connections are a scarce and time-consuming resource, so we don't want to have a new connection for each and every statement to post to the database.
The integration of iBatis is then assured by the container JNDI which locates and supplies such data source.

5.2 – Core Business Logic

The business logic can be developed in another layer of isolation, knowing only the methods provided by the DAO layer, hiding the database complexity. The EJB layer (session beans) provide all the pooling, caching and performance features on the J2EE blueprints.

– **Session Beans:** they encapsulate the business logic as a façade for reading and writing operations, as well as any other business operations.

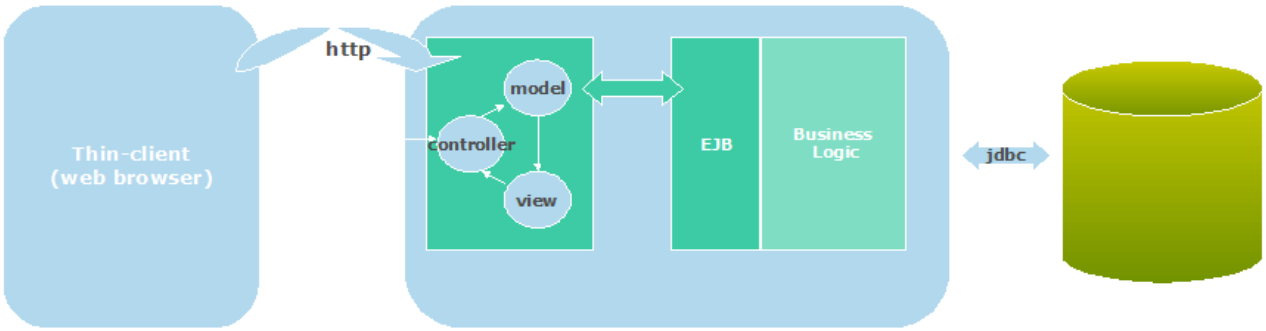
– **DTO (Data Transfer Objects):** The OR/Mapper creates and populates these objects automatically with data from the database. They are an object representation of the tables' structure, and are returned from methods in the DAO.

– **BO (Business-logic Objects):** These are POJOs (Plain-Old Java Objects) that can be developed considering only the DAO layer and the DTO that it supplies, and are encapsulated for usage under EJB mainly.

5.3 - Core Application Workflow (Model-View-Controller)

The web application has a generic workflow mechanism based on the Model-View-Controller (MVC) paradigm. It accepts requests from any presentation layer, validates it, and provides an appropriate response. The framework adopted to implement such paradigm is the Struts Action Framework (http://struts.apache.org) which is widely used and accepted.

The following picture displays the integration between MVC concept and the Net-WMS architecture.



- Fig. 6 -

– **Controller:** this component is a the entry-point for all web requests. Struts already provides a rather complete implementation with features such as monitoring access to resources, roles, etc.

- **Model:** Struts Actions that are invoked by the controller based on requests received. These actions are the integration point with the business logic, namely the EJB layer.

- **Action Forms:** Part of the model, these forms are a representation of all parameters that can be passed in each request, stating data types, and possible validation.

- **Validation:** The Validator Framework (included in the Struts distribution) automatically validates an Action Form according to a set of rules expressed in XML files. This includes all the server-side validation required for “required” fields, “integer”, “email”, “maximum length”, “date”, etc, that forms from the presentation layer may submit.

- **Security:** Container-Managed Security (i.e., managed by the container JBoss) using a predefined realm for LDAP/Active Directory authentication. Application authentication is performed in the Form type.
  In the scope of the development environment only a database authentication is considered.

- **Logging:** all actions from any class in the application, or any class in the open-source frameworks mentioned provide logging to plain-text files. The framework Log4j (http://logging.apache.org) is a “de facto” standard in this chapter.

5.4 - Presentation Layer

The presentation layer encapsulates all design and presentation-specific resources. It is very important that the view is not mixed with any business logic. It is also important to keep the development requirements low in the presentation layer such that any web-designer can read and understand the files. If there is any Java complexity blended in HTML, then it becomes much harder. So we shall stick to plain HTML and tag-libraries in the JSPs.

However, in the spirit of the Web 2.0 technology trend, the AJAX technique may be applied at some key points when appropriate.

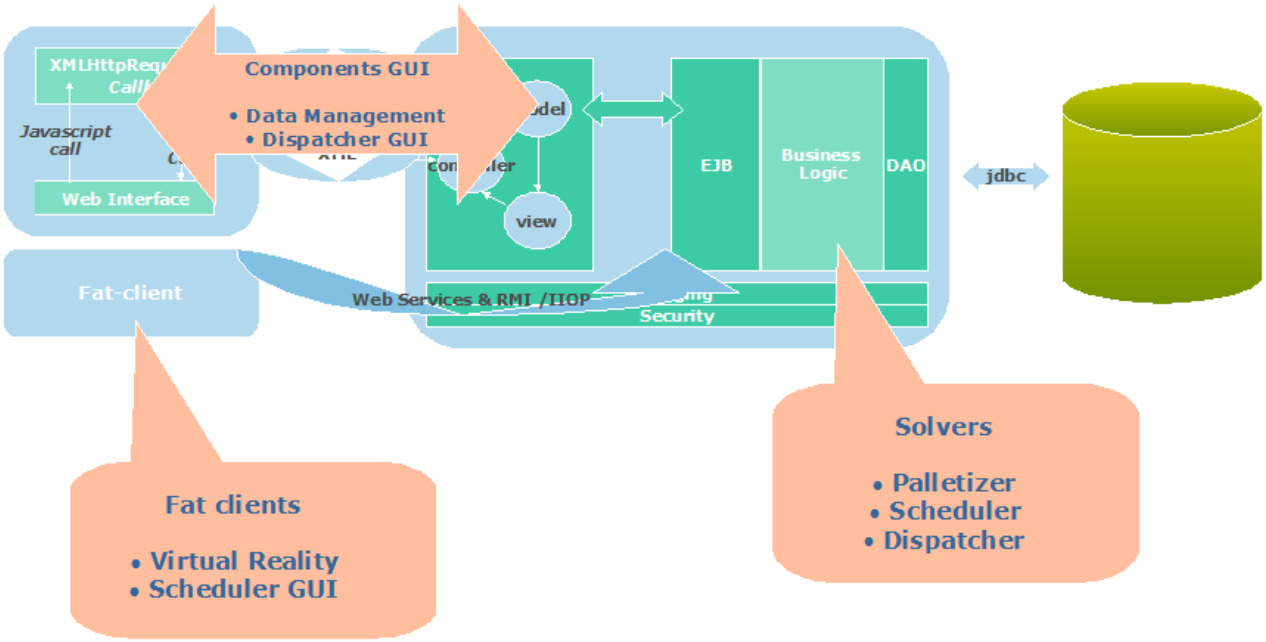
- **JSP:** HTML-based files using images, css, javascript, etc. Dynamic contents should be placed using only tag-libraries for web-designers to easily understand and extend.

- **Tag-Libraries:** Struts provides most of the tag libraries required for many applications such as the “logic”, “html” and “bean” tag-libs, but custom tag-libs can also be developed.

- **AJAX:** the open-source framework Rico (http://www.openrico.org) is adopted as the AJAX engine.

6 – Conclusion

The Net-WMS architecture can then be resumed in the following picture.



- Fig. 7 -

Considering all aspects of Web 2.0 and AJAX, the MVC paradigm, the EJB layer, the Web Services production and the overall Service Oriented Architecture, the Net-WMS architecture encapsulates the solvers (palletizer, scheduler and dispatcher) and provides services on top of them.

Those services feed the Scheduler and Dispatcher GUIs either as thin or fat clients, as well as the virtual reality client. A web front-end enables the data management.

To conclude, the proposed architecture fits Net-WMS requirements for a SOA that promotes Web 2.0 concepts. These requirements satisfy both the networked environment of services and knowledge, as well as the user experience and interactivity.

7 – References

- http://java.sun.com/j2ee/j2ee-1_4-fr-spec.pdf, J2EE specification
- http://msdn.microsoft.com/architecture/soa, Service-Oriented Architecture introduction
- http://www.jboss.com, Jboss Application Server
- http://tomcat.apache.org, Tomcat servlet container
- http://oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html, Web 2.0 introduction
- http://openrico.org, AJAX open-source framework
- http://struts.apache.org, Struts Framework
- http://logging.apache.org, Logging services
- http://ibatis.apache.org, OR-Mapper Ibatis

}