

Decision Support System Software *White Paper*

BY

*Abhijeet Pradhan
Corinne Lee
Deepa Saldanha
Gautam Sampath Kumar
Lei Lai
Min Huo
Refeng Wu
Vorapong Chanawatr*

May 9, 2002

1	Scope of Document	4
2	Client Company Background	4
3	Project Scope.....	4
3.1	Business Context.....	4
3.2	Description of the problem	4
3.2.1	XML/XSL based Website	4
3.2.2	Java Detection Applet	5
3.2.3	Common Comparison Applet	5
4	Objectives.....	5
5	XML/XSLT Website.....	5
5.1	Current System Architecture.....	5
5.2	Conceptual Model for XML-XSL Transformation.....	6
5.3	Proposed Architecture.....	7
5.4	System Development	7
5.4.1	Typical pages selection criteria.....	8
5.4.2	Parser evaluation and identification:.....	8
5.4.3	DTD Development	9
5.4.4	XML Development	9
5.4.5	XSL Development.....	9
6	Common Comparison Module.....	10
6.1	Background	10
6.2	Changing Emphasis	10
6.3	Technical specification and working prototype	11
6.4	Process flow	11
6.4.1	Process flow for group decision making in a distributed environment	11
6.4.2	One-time setup for the group decision process	12
6.4.3	Types of decision-making process	12
6.5	Technical requirements.....	13
6.6	Selection of technology for common comparison	14
6.6.1	JSP / Servlets.....	14
6.6.2	Applets	14
6.6.3	Flash MX vs SVG.....	14
6.7	Design of prototype.....	15
6.8	Common comparison module components.....	15
6.9	Known issues with common comparison module prototype	16
7	Browser/OS/JVM Version Detection	16
7.1	System Architecture	16
7.1.1	Detection utility.....	16
7.1.2	System administrator panel.....	16
7.2	System Design.....	19
7.2.1	Requirement Gathering.....	19
7.2.2	Requirements Document.....	19
7.2.3	Technology Decision	19
7.2.4	System Design.....	20
8	Decision making in a distributed environment	20
8.1	An Analysis of Microsoft .NET.....	21

8.1.1	Description of .NET Framework	21
8.1.2	.NET My Services.....	22
8.1.3	The client's software use of the .NET Framework	23
8.2	An Analysis of Sun's J2EE Architecture	23
8.2.1	Description of J2EE Architecture	23
8.3	Comparative Analysis of the Frameworks.....	27
9	Migration plan to J2EE	32
9.1	Description of J2EE Architecture	32
9.2	Current Architecture of the Decision Support System	33
9.3	Proposed J2EE architecture for the Decision Support System	35
9.3.1	Important Features of the architecture	35
9.3.2	GUI.....	36
9.3.3	JSP/Servlets.....	36
9.3.4	Enterprise Java Beans	36
9.4	System Components:	37
9.4.1	J2EE Server.....	37
9.4.2	Database	39
9.4.3	Recommendation	40
9.4.4	Proposed implementation time line	41
10	Marketing Plan.....	42
10.1	Product and Service Offered	42
10.2	Target Market.....	43
10.3	Positioning	44
10.4	Marketing	44
10.4.1	Partnerships	44
10.4.2	Potential Partners	45
10.4.3	Pricing.....	45
11	Reflections / Conclusions	46

1 Scope of Document

This document describes the work that the team has done. The client, Decision Coaches, Inc. and its product, AliahTHINK! v5.0, will be referred to as "the client" and "the product" respectively throughout this document. We will document the technical considerations and the technologies employed in the implementation of the solutions that we gave to the client. We also will share the business models as well as the marketing plan.

2 Client Company Background

The client is a Pittsburgh-based software firm that specializes in building Decision support software based on the Analytical Hierarchy Process (AHP). The client's product is a decision support system that facilitates individual and group decision-making processes. It takes inputs from the individual/group on weighing various criteria against one another, and proceeds to apply AHP algorithms to those inputs to display the results to users in a numbers of forms. It is a desktop application and is usually accompanied by guided decision-making by a coach. It caters to individual decision-making as well as group decision-making.

3 Project Scope

The project scope has been mapped to the overall business context, the business issues and the definition of the problem statement.

3.1 Business Context

The client focuses on helping clients translate business priorities to actions through the use of advanced decision-making technology. The advances in technology and the increasing use of the Internet have resulted in an explosion in the availability of information one of the direct consequences of which is the increasing complexity of the world in which we operate. Trends in the use of the Internet indicate that the next big Internet wave will involve the connecting of decision-making systems of one place with that of another.

The client is well positioned to take advantage of this trend but needs to migrate and enhance its existing application, as appropriate, to deliver platform independence, deployment flexibility and user application flexibility.

3.2 Description of the problem

3.2.1 XML/XSL based Website

The client website currently utilizes a series of HTML pages housed in a database, Sybase Adaptive Server Anywhere, and is deployed through Sybase PowerDynamo to Microsoft Internet Information Server 4.0. This architecture was implemented to address concerns regarding security and per page access time. The client now desires to translate this existing website into an XML/XSL structure, providing data access to the entire the client website and reducing change-management issues regarding website modifications while preserving or improving upon the above stated concerns.

3.2.2 Java Detection Applet

The client's decision support software uses Java applets that require a minimum Java Virtual Machine version when deployed to Microsoft Internet Explorer. Microsoft does not, by default upgrade the JVM. Therefore, a mechanism of some sort is necessary for detecting the whether or not Java enabled on the client-browser, whether the presently installed version of the JVM meets the specified minimum requirements and responding as appropriate.

3.2.3 Common Comparison Applet

At the heart of the client decision-making process is the "Comparison" in which one or more users compare the criteria, both quantitative and qualitative, of their decision against each other. While the existing applet accommodates a pair wise comparison, allows it's appearance to be customized to a degree and it's "flow" somewhat tailored to the needs of the e-Decision it does not provide for multiple users to contribute their "vote " nor does it provide for alternative GUIs for comparison. Therefore, an interactive mechanism is required to enhance the online Comparison functionality.

4 Objectives

The objectives and the scope of the project are as follows:

- /// To deliver working prototypes of three individual components that have been thoroughly tested and documented, to the client. The three components are
 - /// A working XML/XSL prototype website based on their HTML website.
 - /// A Java detector that will retrieve important information on a client browser's Java capabilities.
 - /// A common comparison GUI that will feature an intuitive and user-friendly interface to enable the user to specify criteria to make decisions.
- /// To conduct a study and analyze the current technology infrastructure that the client uses to deliver its products and solutions to its clients.
 - /// Analyze the different frameworks available for distributed applications
 - /// Recommend a suitable framework for the client's requirements.
 - /// Develop a migration strategy to deploy the client's decision support system in the recommended framework.
- /// In addition, the team has also developed a positioning strategy that can be used by the client in the overall marketing plan.

5 XML/XSLT Website

5.1 Current System Architecture

The main task is to translate existing HTML based the client's website into a XML/XSL based website. The present website is housed in the database system of Sybase Adaptive Server Anywhere and deployed through Sybase PowerDynamo to Microsoft Internet Information Server 4.0. The HTML page then is displayed on the client web browser while requested via Internet. The current system architecture is as shown in Figure 1.

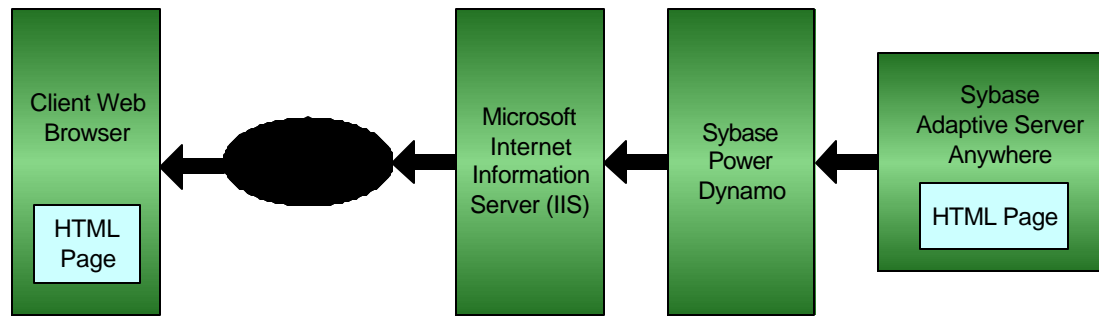


Figure 1: Present Client's Website's Architecture

XML is an extensible language used to structure and describe the data that needs to be presented. XML will be as important to the future of the Web as HTML has been to the foundation to the web. A large number of software vendors have adopted XML as the standard. The client is about to accept this standard as well for future development.

By translating the HTML based website into the XML/XSL based website, the company will be able to separate data from presentation. This will enable the company to easily change the underlying data without worrying too much as to how it will be presented on the web. This will reduce the cost of making modifications on the website.

5.2 Conceptual Model for XML-XSL Transformation

The following diagram shows the conceptual model, which is the basis of the system design and thereby, the system architecture for the website.

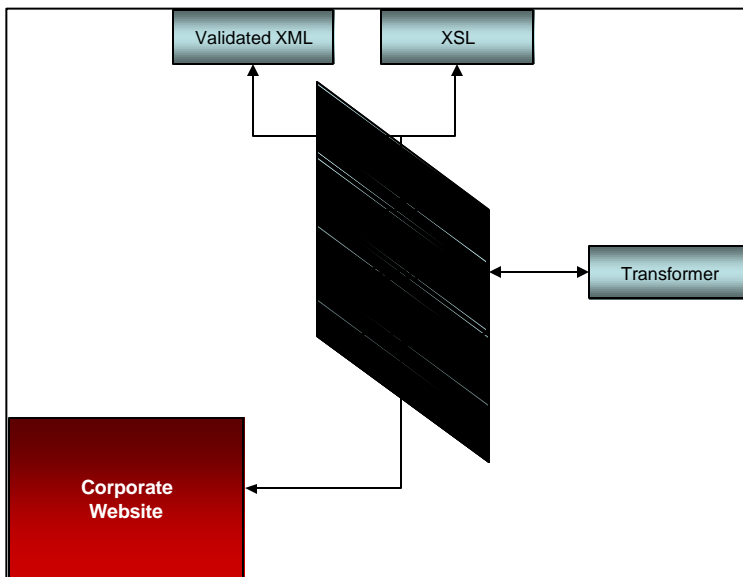


Figure 2: Conceptual design of the proposed client's corporate website.

The above diagram gives a conceptual design of the proposed client corporate website. The corporate website will be generated through a Servlet, which is invoked when the user enters the corresponding URL in the browser. In response to this request the Servlet retrieves the XML and XSL code, in the form of bit streams, for generation of the corresponding HTML page. It then sends this information to the transformer, which then sends the HTML code back to the Servlet. The Servlet then returns the HTML page to the browser in the form of an output stream.

5.3 Proposed Architecture

Given that we are using Servlets in the proposed architecture, a change would be required from the current system architecture to be able to handle Servlets. Microsoft's IIS cannot handle Servlets and a Servlet container would be required to be able to manage them.

It is for this reason that we propose Apache's Tomcat, which serves the dual purpose of being a servlet container and a web server.

The proposed XML/XSL based website would have a similar data flow regarding the page extraction. Data is stored in XML format in the Sybase database. All the data is stored in XML format in the Sybase Adaptive Server Anywhere and deployed by Sybase Power Dynamo. Before the data is sent to Tomcat Server where it is available on the Internet, an XML/XSL transformation engine is needed to transform the XML pages into HTML pages based on XSL pages

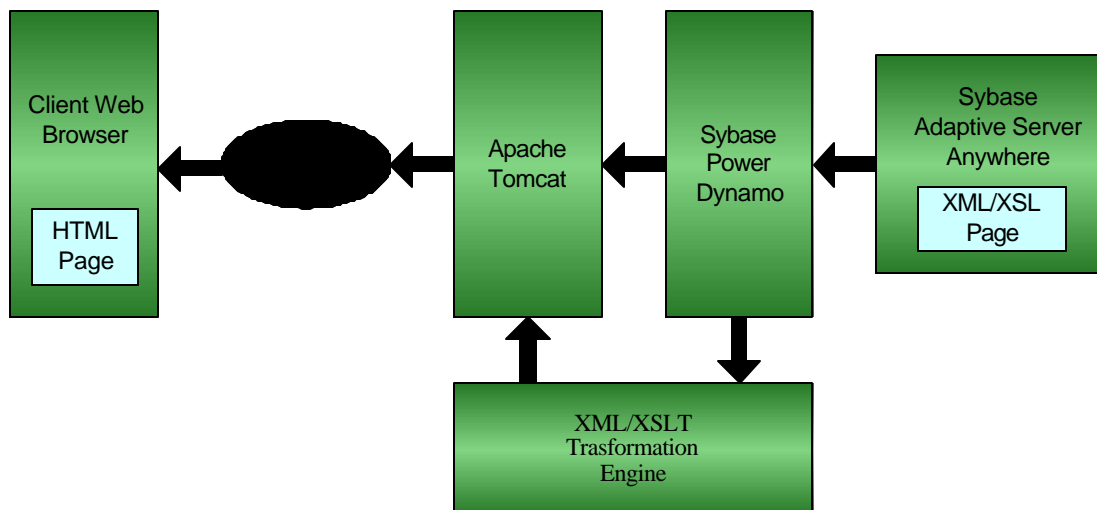


Figure 3: Proposed Client's Website's Architecture

This engine will retrieve appropriate XML and XSL pages based on the request from users, transform XML into HTML using XSL transformation, and then send the HTML pages to the Internet via Tomcat Server.

5.4 System Development

The figure below depicts the overall process and methodology followed by the team to achieve the required objective.

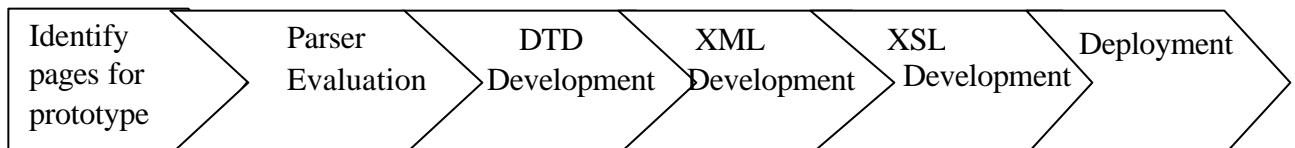


Figure 4: XML\XSL System Development Overview

For the development of the XML/XSL based website, the first step was making the Document Type Definition (DTD) file which defines the XML document structure with a list of legal elements.

This was done based on the content of the pages of the existing corporate website. Also, this DTD is aligned with the current client's customized XML specification.

Next steps were developing the XSL that would deliver the desired HTML pages. A subset of the HTML pages on the corporate website was chosen for development of the prototype.

5.4.1 Typical pages selection criteria

As was decided at the time of the mid term presentation, we chose typical HTML pages for the prototype and transformed these pages. The team set three criteria to select the typical pages on the current website.

The pages should cover each major component of the website. In general, the homepage is the index for all pages, and there are three major components for the system, online application, solutions - products and services, and decision processes. So we select homepage, and at least one page from each of the components.

The pages should cover the most elements on the website. The typical elements on the website includes: text, image, linkage and form. The homepage includes almost all the elements. Some of the pages are characterized more for the images, while others are based on the text elements. And we set a typical form page too. For each of the typical page, there is a format for kinds of linkages located on the top section and left section of the pages.

5.4.2 Parser evaluation and identification:

The parser is one of the most important layers to any XML-aware application. This component handles the important task of taking a raw XML document as input, ensuring that it is well formed and if a DTD is referenced, a validating parser is able to ensure that the document is valid. The result of this procedure is a data structure that can be manipulated and handled by other XML tools or Java API's.

We evaluated eight transformation parsers available on the market. There are: Microsoft XML Parser, IBM Alphaworks XML Parser for Java, Apache Xerces Java, James C. Clark's expat (XML Parser Toolkit) and XP, HP labs' HEX (The HTML Enabled XML Parser), Sun Microsystems Java Project X, and Oracle XML Parser. Most of them are free.

We evaluated the transformation parsers based on their conformance to the XML standard. A test suite has been defined by the Organization for Advancement of Structured Information Systems (OASIS). The suite contains a set of over 1000 valid and invalid documents to check a parser's capabilities of accepting the valid ones and rejecting the invalid ones.

We recommend the use of Xerces as the website parser based on the following reasons.

- /// It is a validating parser – The requirements for the project state that an XML application language for existing HTML content must be defined and used to validate the documents.
- /// Xerces enables on-the-fly validation of a document – Therefore, rapid deployment of the web pages will be maintained.
- /// It will maintain the present architecture afforded by Adaptive Server Anywhere.
- /// Xerces is freely downloadable therefore; cost in terms of purchasing the software product is not incurred. Also, the learning curve is not steep.
- /// Xerces is popular in the industry.
- /// Xerces-J is designed for use on the Java platform – this will help it to be integrated with the rest of the application.

5.4.3 DTD Development

The single DTD was developed by studying the content of all the pages of the client's corporate website. DTD tags were identified on the basis of the type of content that the pages contained. The tags enabled storing information like the page title, page description and contents of the sub-page. Most pages on the website followed the standard format of having a page title, a general description of the page and more specific content in the rest of the page. This specific content fell into the sub-page category. This content was displayed on the pages in a non-standard format. The DTD accommodates for this, to enable making a replica of the corporate website, with the use of additional tags. Other pages on the website included forms for collection of customer data. The DTD includes tags for these pages as well.

Once the main structure of the DTD was identified, the DTD was broken into several parts for easier understanding and for making the design modular; using external parameter entities and external parameter entity references.

5.4.4 XML Development

Validating a document is like checking a program for syntax errors. Only after a program is checked can the compiler perform its main job of generating code. And only after a document is validated can a program easily process the data within it. An XSLT programmer, for example, relies heavily upon the fact that an input document is of a certain structure.

XML files were developed for the content of the selected subset of HTML pages from the corporate website. These files were validated against the DTD for the website with the help of a validation program. The method of validation employed in the program is based on SAX. Any changes made to the XML files will have to be validated against their respective DTDs with the help of this validation program.

5.4.5 XSL Development

XSL was developed to deploy the content of the XML files in the desired HTML format; which is similar to the current corporate website. XSL was for each of the HTML pages in the subset of HTML pages from the corporate website.

5.4.5.1 *Deployment of the system*

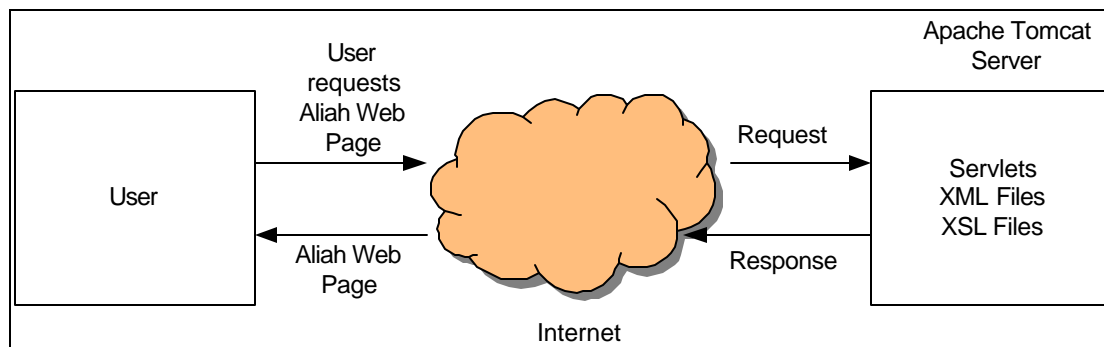


Figure 5: Deployment of the proposed system

The above-mentioned code must be deployed on the Apache's tomcat server. Assuming that the server is installed on the computer, the following steps have to be carried out in order to install and be able to generate HTML on the fly.

The XML and XSL files must be stored in the webapps folder in the tomcat directory. The servlet is stored and its class files are to be stored in the classes folder where the tomcat server is installed.

6 Common Comparison Module

6.1 Background

The common comparison user interface is at the heart of the client software's decision-making process. The common comparison gives the decision maker an easy-to-use intuitive interface to weigh various criteria against one another. At the end of this process, decision algorithms based on the AHP (analytical hierarchy process) process the information inputted by the user and give him needed information on which to base his decisions.

The common comparison interface is currently part of the client's Desktop application, and is therefore not suited to a distributed decision making scenario.

The objectives of this module were

- ✍ To port the current common comparison interface to a web-enabled distributed application.
- ✍ To enable the common comparison to be peer-to-peer between clients. Every users would know where other users are in the decision making process.
- ✍ To render every component of the graphical user interface generic. The common comparison GUI had to render/build itself completely from an XML-based configuration file that would specify various characteristics of the GUI such as width, height, image locations etc. This would give non-technical users the ability to modify the look-and-feel of the interface by simply editing the XML configuration file.

The business benefits of implementing the common comparison as described above are:

- ✍ To enable users in locations distributed across the world to participate in a decision making process using the distributed common comparison module.
- ✍ To simulate the real-life turn based approach to decision making in a distributed web-based environment.
- ✍ To provide the client's marketing department an easy means to modify the look-and-feel of the common comparison to suit customer tastes and preferences, freeing up the technical personnel to focus on core technical development of the client's Application.

6.2 Changing Emphasis

The initial emphasis of this module was to analyze every component of the existing common comparison GUI that was also available as a Java applet, to define a generic means to build this interface based on XML. However, after a meeting with the clients, where a decision making process was simulated and presented to the team, the significance of the peer-to-peer requirement of the common comparison interface emerged. This was crucial to implement a take-turns approach wherein each client would take turns weighting criteria.

Note: Peer-to-peer, in this case, does not refer to the ability for clients to function without a central server, but to the ability of every client in the process to know where other clients are in the decision making process.

6.3 Technical specification and working prototype

Given the complexity and the evolving requirements for the common comparison interface the team felt that the best approach would be to document the requirements and technical specifications detailing the capabilities of the common comparison interface and implement as much of these requirements as possible in a working prototype. The current prototype implements much of the needed functionality of the technical specification, except for some optimizations needed for the common comparison to perform better, and a few of the components are not as generic as specified in the technical specification.

Also, the take-turns based approach is currently implemented using a database polling mechanism, wherein each client polls the database periodically to know where their peers are in the decision making process. A more suitable mechanism would be to implement server push technology wherein the server pushes information to the client. This will obviate the need for database polling. The stage of implementation of the prototype in terms of the requirements outlined in the technical specification will be discussed in detail in a subsequent section.

6.4 Process flow

Our next step was to come up with a business process flow of the common comparison, in order to better understand the business requirements of this module and then map these business requirements to technical requirements subsequently.

6.4.1 Process flow for group decision making in a distributed environment

The team met with the client senior management, for a simulation exercise of the actual decision making process. The exercise has given the team a number of insights into the rich interactions that happens between players and the value of these interactions in the decision making process. These interactions could influence the outcomes in decisions.

We have documented in detail the process flow of the decision-making, as well as the technical specification in the sections below.

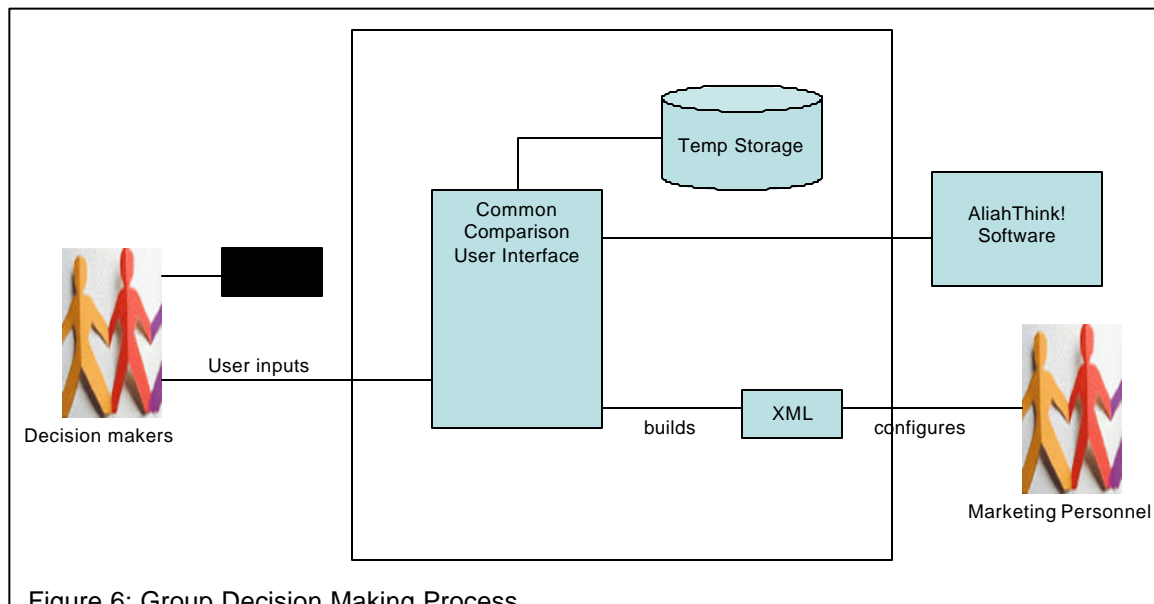


Figure 6: Group Decision Making Process

6.4.2 One-time setup for the group decision process

There is a one-time setup needed for the group decision process. This will be done by the client. After gathering the group decision process requirements from the customer, he will set up the parameters associated with this decision. Examples of these types of parameters are the desired graphics, color, font sizes, voter names, voter ids, etc. He will setup these parameters in an XML-based configuration file. The objective of this process is to customize the look and feel of the common comparison user interface for that particular project / decision.

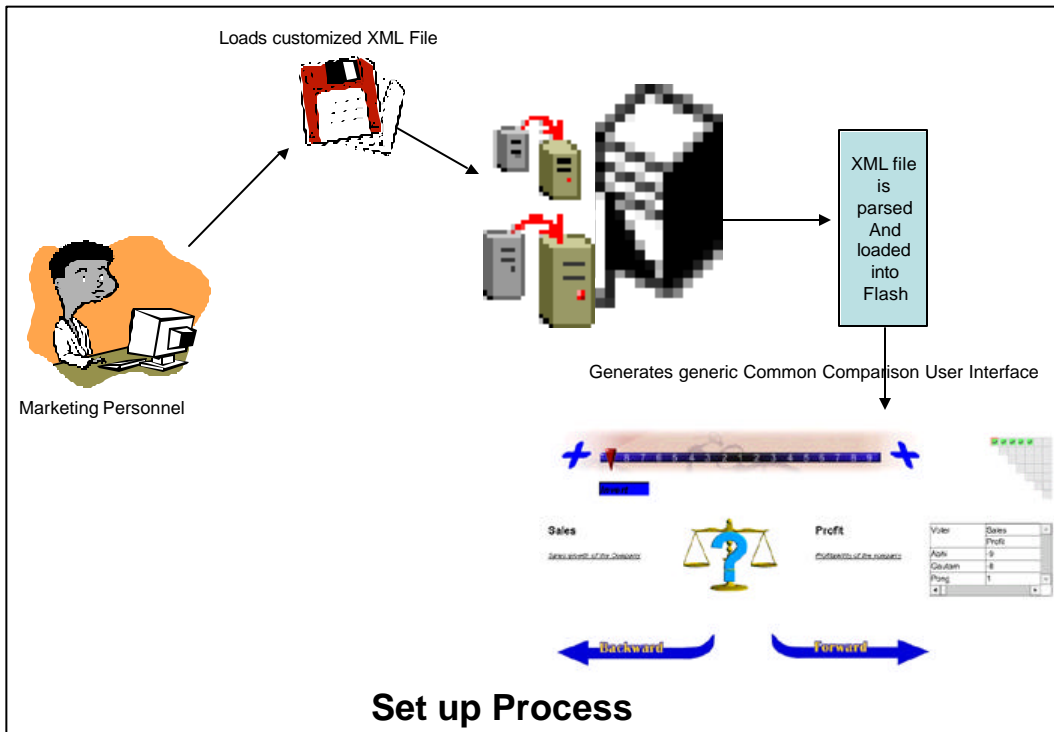


Figure 7: Setup Process

6.4.3 Types of decision-making process

Individual users in the group get turns to set precedence of one criterion over the other. In the decision making process that can be rendered in a distributed environment, there are two possible ways in which the user could give inputs for the decision making process: Decision making process happens as a group but in a distributed environment. Users are allowed to set precedence in a take-turns based approach. Decision making process happens as individuals in a distributed environment. In this case, users are specified with criteria for which they are responsible to set precedence. Users log in at their convenience, set the precedence for criteria that they are responsible for. If other users who are responsible for setting criteria have not yet done so, the input fields are disabled and these fields cannot be populated. Only when a user has set the precedence that he is responsible for can the other users give their inputs for those parameters.

The above types of decision-making are depicted in two scenarios:

6.4.3.1 Scenario 1: Take-turns approach

Let's suppose there are 3 users involved in the decision-making process, and they are user1, user2, and user3.

There are 4 criteria: growth, sales, profits, and market size
There's a comparison between 2 criterions at a time.

In a group decision support setting, users take turns to decide the priority of the criteria and to put their inputs.

- ✍ User authentication is done by the client's decision support software when the user logs into the system
- ✍ Based on the user id, the project id the common comparison user interface will pull up the corresponding customized user interface for that decision/project.
- ✍ On the user interface, he will see the other users and their status, whether they have logged into the system or not. Before all the users have logged in, the process is disabled and users will have to wait until everyone is logged in.
- ✍ Only when all the users have logged in, then the decision process can begin.
- ✍ User 1 will be given the option to select which criterion is more important. After User 1, User 2 and then User 3 will weigh the same criteria.
- ✍ For the next pair of criteria, User 2 will decide which criterion is more important. Then User 3 followed by User 1 will enter their weight-values.
- ✍ This take-turns based process continues until all criteria have been weighed.

6.4.3.2 Scenario 2: Distributed asynchronous approach

This will be more a distributed decision model, where users can log on, in their own time and key in their inputs.

User 1 goes into the system, and all the Pairwise criterions that are assigned to him are waiting for him to input. User 1 does not need to wait for User 2 or User 3. Each user can work on the system and key in their inputs without tying in with the other users.

The advantage of such a model is the flexibility in time; there is no need for concurrency or taking turns. However, this approach tends to dilute the interaction, collaboration of a group decision-making process and the richness it brings to the decision. Although this scenario truly exploits the power of a distributed system, the richness of the actual group decision-making process is diluted. Also, strategically it is important that the decision coach from the company facilitates the decision making process since this will demonstrate the value added by the client. This model has been implemented in the prototype as an option that may be used by the client or its customers at their discretion.

6.5 Technical requirements

After understanding the business requirements of the common comparison applet we proceeded to come up with a set of technical requirements that we felt the common comparison applet had to satisfy to fulfill these business requirements.

- ✍ **Web-enabled:** The common comparison GUI had to be web-based, as this would lend itself to a distributed decision making setup. To make the GUI web-enabled essentially web-based technologies would need to be used to implement the client using web-based technologies.
- ✍ **Generic implementation of interface:** A critical requirement of the common comparison GUI was to implement the GUI and it's various components using as generic a means as

possible. This would be done using an XML-based configuration file that would specify in minute detail the look-and-feel as well as behavior of the common comparison GUI.

- ✍ **Temporary storage of inputs:** The inputs of the users i.e. The values assigned to various criteria will have to be stored temporarily. This will enable users to move back and forth between previously weighed criteria. This will also serve as a base for implementing the peer-to-peer architecture described in the next requirement. The temporary storage will be implemented in a database.
- ✍ **Peer-to-peer architecture:** Each client will have to know at every point of time, the status of other clients in a visual display, so that every client knows the current stage of the decision making process. This is crucial to implement a take turns based approach, wherein one client has to wait for another client to weigh a particular criteria against another, before proceeding with the decision making process.
 - **Database polling:** As a result of the peer-to-peer requirement, the additional technical requirement of polling the database periodically arises. A database is being used for temporary storage of user inputs. By polling this database periodically, each client is able to update it's visual display to depict the current stage of the decision making process in a color-coded matrix that will be described in detail in a subsequent section.

6.6 Selection of technology for common comparison

After clarifying our technical requirements, we proceeded to evaluate a set of technologies against these requirements to determine which technology would best solve the technical requirements outlined above.

6.6.1 JSP / Servlets

We evaluated using a purely server-side driven approach to solving the problem as the application logic would be closely tied to the database and we would be in a better position to implement a server-push technology. However, JSPs and Servlets that would be used to generate html pages were ill suited to the kind of sophisticated graphical components such as drag-able sliders, matrices and view tables that we had envisioned for the final application

6.6.2 Applets

It was apparent that we needed to have server side technology such as JSPs and Servlets to tie the presentation layer to the business logic, and also implement the application logic. However, for reasons outlined above we decided that a purely server driven approach was not suitable. We evaluated using Java applet technology on the client side.

However after careful consideration of the effort involved in building generic graphical components that would have sophisticated behavior, we decided that Java applet technology was not the most suitable. Also, after a discussion with Aaron Wollerton we decided to take the concept of "genericity" a step further to extend the actual components which were either text-based or static graphic images to include support for multimedia and audio components.

6.6.3 Flash MX vs SVG

Based on the technical requirements we zeroed in on 2 technologies that we believed were well suited to build the client side of the common comparison module. These were Flash MX technology from Macromedia, inc. and scalable vector graphics (SVG) from Adobe Corporation. We decided to use Macromedia Flash for the following reasons:

- ✍ Flash technology has been longer in the market than SVG and hence there are more implementations of superior front-end user interface using Flash than SVG.
- ✍ The popularity of Flash also gives developers more resources on the Internet or books to solve technical issues.
- ✍ Flash technology is proven.
- ✍ SVG is newer, and it has gained popularity only in the past 6 months.

6.7 Design of prototype

After we selected Java Servlets for the server component and Flash MX for the client component of the common comparison module we proceed to design a prototype that would satisfy the technical requirements listed earlier. The architecture of the common comparison module is depicted in Figure 8 below.

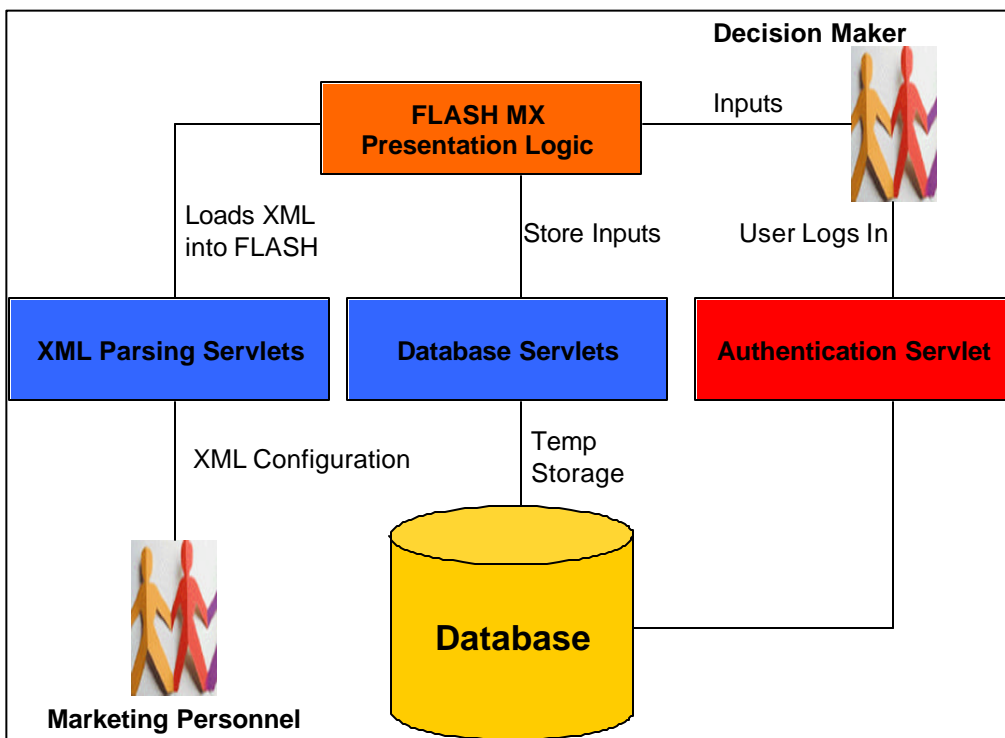


Figure 8: System Architecture

6.8 Common comparison module components

The core of the common comparison module is a flash movie. This contains all the logic written in Macromedia's scripting language ActionScript to render all Graphical User Interface components based on parameters specified within an XML configuration file. The various tags/components of the XML file are related to the parameters for the Comparison, Criteria, Participants, graphic components and other details.

6.9 Known issues with common comparison module prototype

There are a few known issues with the current implementation of the common comparison module prototype:

- ✍ As outlined above all tags have not been implemented completely in the current common comparison module. These need to be implemented to bring full functionality to the module.
- ✍ The Flash prototype performs reasonably well in the Macromedia Flash test environment but there are noticeable performance issues when the Flash GUI is deployed to a browser. Response times are slower to mouse clicks for instance. Any known performance issues with Macromedia on the browser will have to be researched. Also code optimizations will have to be carried out. The fact that this is a prototype environment, using a lightweight web browser (tomcat) and a lightweight database (MS-Access) may have impact on the performance of the prototype as well.
- ✍ Flash and RMI: future extensions suggested by us in the migration strategy document include the use of Java remote method invocation (RMI) callback technology to implement server push technology. This will enable all clients to know where their peers are in the decision making process. Currently, there is no known API/ interface between Flash and Java RMI. However alternatives such as using Flash to talk to a Servlet that is RMI-enabled could be explored.

7 Browser/OS/JVM Version Detection

7.1 System Architecture

The java detector is a stand-alone web-based application. The Application can be divided into two distinct parts.

- ✍ A software utility that performs a check on the client's computers to determine the version of browser, operating system and Java virtual machine that they are running.
- ✍ A system administrator panel to configure the back-end database with the minimum requirements.

7.1.1 Detection utility

The application detects the client's browser type, version, JVM version and operating system type. This is compared with the minimum system requirements stored in the back-end database. If client's browser, JVM and OS meets the minimum requirements, the client could access the client's software, i.e. Common Comparison Application; otherwise, the client will be advised to download newer versions (possibly types) of browser, JVM or OS.

7.1.2 System administrator panel

The system enables the system administrator to access the back-end database and configure the minimum system requirements.

The following Figure 10 is an architecture diagram for using JavaScript (HTML), an applet, and a servlet to detect client browser, JVM and operation system.

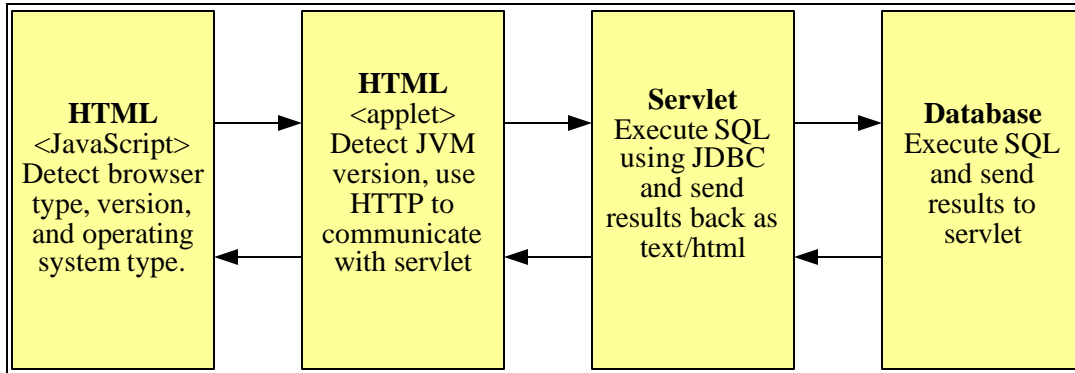


Figure 10: JavaScript – Applet – Servlet – Database Architecture

The following figure 11 is an architecture enable runtime configuration of mini system requirements in the back-end database used by system admin.

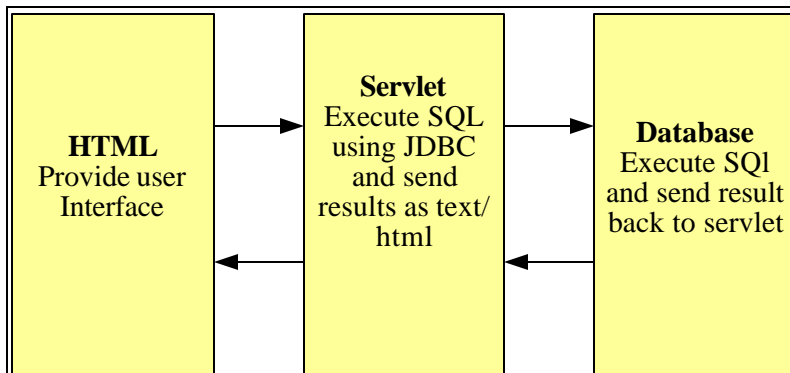


Figure 11: HTML – Servlet – Database Architecture

The following flowcharts highlight what happens when users access the system.

Chart 1: Detection Part

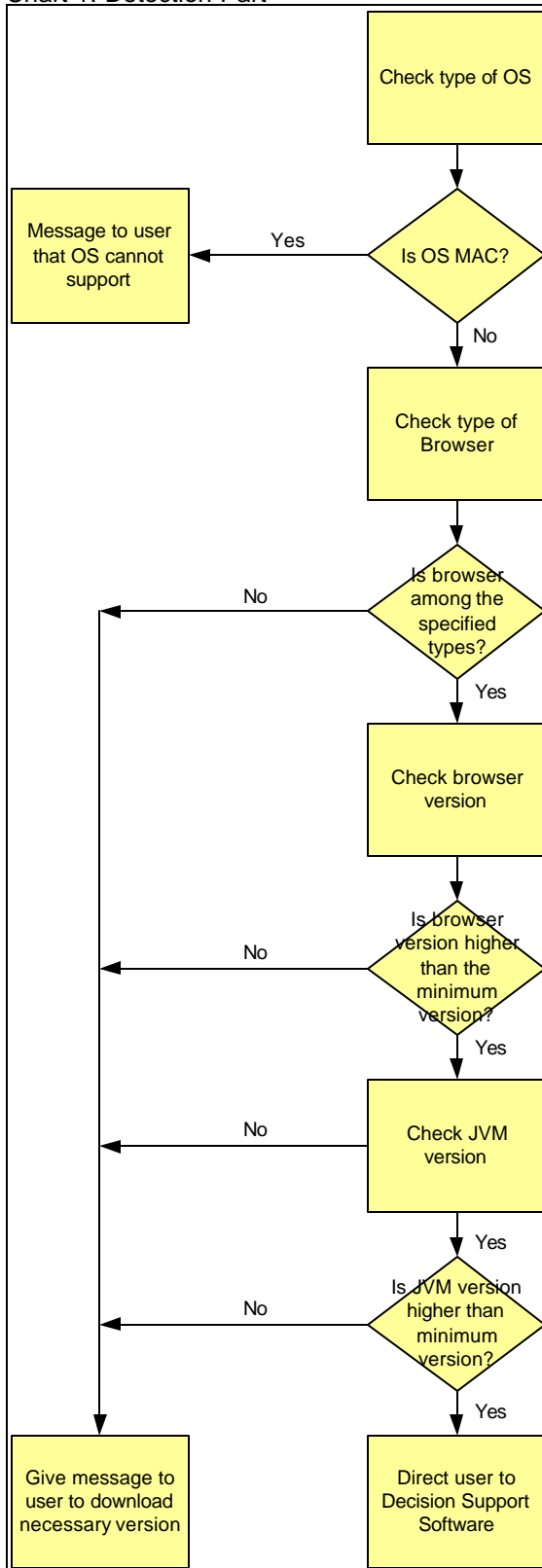
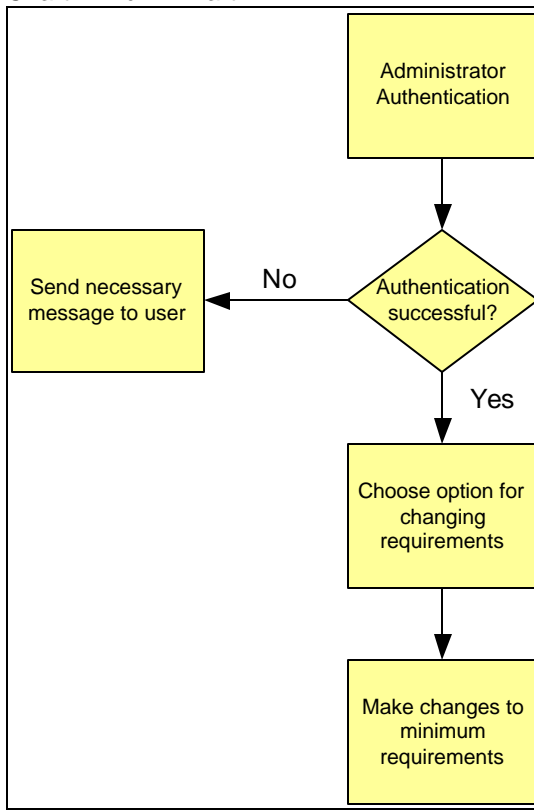


Chart 2: Admin Part



7.2 System Design

In this part, we will explain the development of the systems based on the following sequence that we followed.

7.2.1 Requirement Gathering

The team met with the client on a number of occasions and understood the business model, products and company. Over a number of interviews over a period of time, the business requirements for the system were gathered.

7.2.2 Requirements Document

Based on the requirements we gathered, the team developed a "Requirements Document". This document covers the business requirements and the technical requirements.

7.2.3 Technology Decision

In order to detect the OS, browser and JVM from the client machine whenever client visits the client's website, we decide to choose a technology which could possibly run on client-side with short response time and higher accuracy. After doing a research on available technologies, we find that if using java code, we may detect browser type and version, but if the browser is not IE and Netscape, we may get false information. So simple java codes cannot accurately detect

browser. Then we find the combination of JavaScript and java applet could be a perfect match on the requirements of accurate detections and short response time. The architecture of JavaScript – java Applet – Servlet – Database was chosen.

7.2.4 System Design

In this section, we will provide a detailed discussion on system design on class-level. We will also provide class diagrams and use case diagrams as part of the system analysis.

7.2.4.1 The detection utility

When user accesses the application, the DetectionApplet.html will be displayed. It will notify user that the OS, browser and JVM of his/her machine will be detected. When the user clicks OK, the JavaScript code embedded in DetectionApplet.html will run. The JavaScript codes will detect the OS type, browser type and browser version on the client side; it will also call a Java applet named DetectionApplet.java. The applet will detect JVM version, then send detected results of OS, browser and JVM to servlet named DetectionServlet.java.

The servlet will connect to a back-end database and retrieve minimum system requirements from the database, make a comparison between the detected data from the client's machine and the minimum requirements from database, and send appropriate recommendations to applet. Applet will receive the recommendation and output the recommendation in HTML format for the browser. Based on the recommendation, user could access the client's software or user needs to download newer versions of browser or JVM, depending on difference between the user machine's configuration and minimum requirements.

7.2.4.2 The administrator panel

The Admin will first be authenticated to the system. The admin is then displayed a menu page, generated dynamically by a servlet called Admin.java. Admin will have four options to change browser version, JVM version, target URL for downloading new versions, and customized recommendation messages. The admin can choose one of them, and a corresponding html page will be generated by servlet according to admin's choice. Once admin input the data he/she wants to change, the back-end database will be updated.

7.2.4.3 Development and Testing

We set up the back-end database first, developed the detection utility and the administrative panel of the application simultaneously, and then integrated them into one application. We tested on UNIX, Windows 2000/NT/XP platforms; we also tested on various versions of Internet explorer and Netscape. Further tests need to be done for other browser types, i.e. konqueror, omniweb, and iCab.

8 Decision making in a distributed environment

The client's decision support system is currently a standalone, desktop application. While this is a powerful application and attempts to capture the wisdom of the people using the tool, it is faced with certain limitation of being a desktop application. In today's networked world, organizations and decision-making is more de-centralized. Location is not a constraint for organizations any more. Executives need to communicate with their counterparts in other locations. Business processes and decision making, therefore, are becoming more distributed than ever. Most of the business applications now are required to meet this requirement from their customers. It is important that the client should have a product that would meet these requirements from their customers.

It was with this objective that the team looked into the aspects of migrating the decision support system as a distributed application. The team analyzed the following frameworks for decision making in a distributed environment: .NET, J2EE

8.1 An Analysis of Microsoft .NET

8.1.1 Description of .NET Framework

The .NET Framework is Microsoft's new computing platform that simplifies Application development in the distributed environment of the Internet.

The various features of the .NET Framework are:

- ✍ To provide a consistent Object-oriented environment regardless of whether an object is stored and executed locally or remotely.
- ✍ To enable developers to develop once and deploy their solutions across multiple platforms and devices using XML-based communication.
- ✍ To allow the development of XML Web services. XML Services allows the development of components that can be used as building blocks for other applications. These components are made available on the Internet as Web Services, listed using UDDI (Universal Description Discovery and Integration), defined using WSDL (Web Services Description Language), and communicated using the SOAP (Simple Object Access Protocol).

Applications are managed or unmanaged under the .net framework and the various components of managed and unmanaged applications are depicted in figure 12 below:

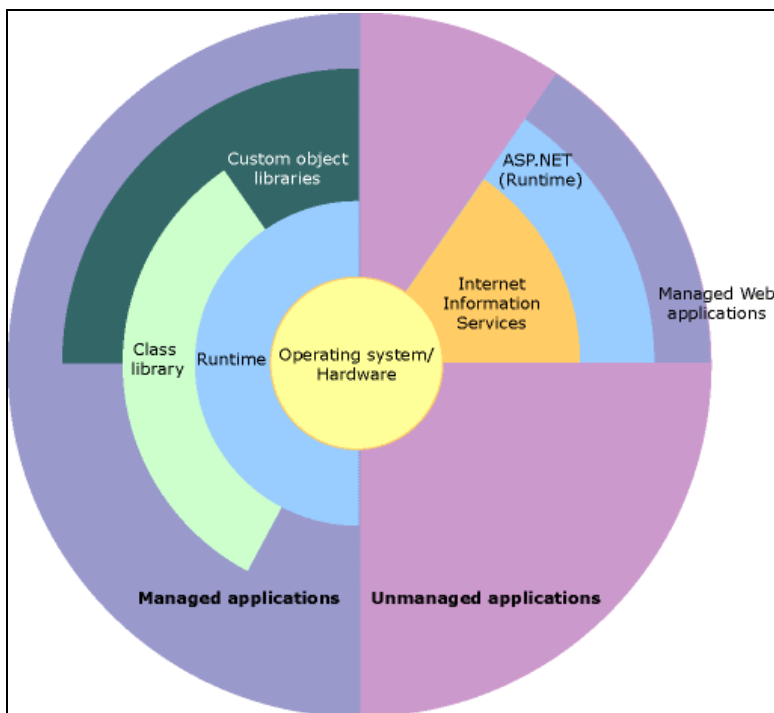


Figure 12: Components of managed and un-managed applications.

The figure 13 below depicts a Basic network schema with managed code running under different server environments. This allows you to use all the features of the common language runtime while gaining the performance and scalability of the host server.

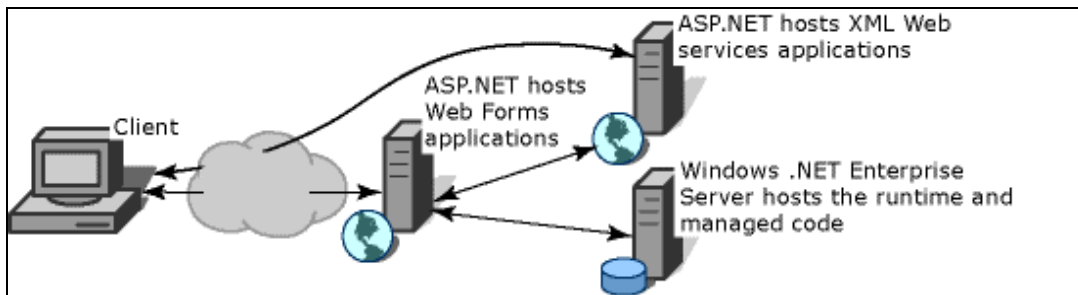


Figure 13: Basic network schema

Servers such as Internet Information Server (IIS) and SQL Server can perform their standard operations, while application logic executes through the managed code.

8.1.2 .NET My Services

.NET My Services are a set of consumer-focused XML Web services that developers can leverage due to the data-centered nature of XML to build Rich applications for disparate Web Sites, Web Services, Platforms and Devices.

The various features of .NET My Services are as follows:

- ✎ **User Centric Web Services:** .NET My Services was developed to focus on User Data and not on specific applications, platforms or devices. This will allow developers to build a wide variety of XML Web Services taking advantages of .NET My Services' user data centrality.
- ✎ **Open Standards:** Transferring data between applications is done using XML Web services and the interpretation and presentation of transferred data is left to the platform- and device-specific applications. All services are accessed by sending SOAP messages containing the XML Data over HTTP.
- ✎ **Passport:** .NET My Services uses Microsoft Passport to authenticate users to use the appropriate level of a Web Service depending on their profile and access rights.

Figure 14 below is an illustration of how clients access user data using .NET My Services.

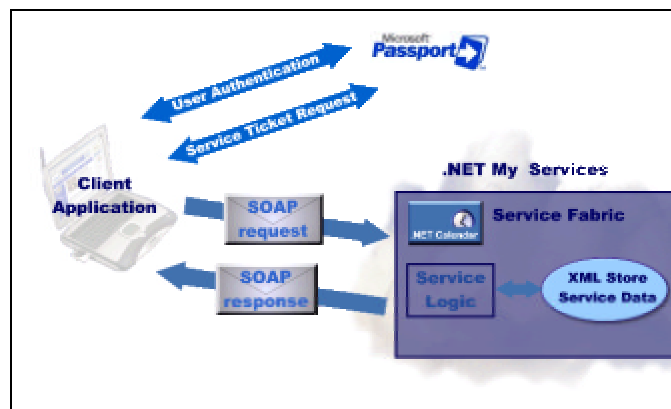


Figure 14: illustration of how clients access user data using .NET My Services.

8.1.3 The client's software use of the .NET Framework

Porting the decision support system software to the distributed .NET Framework will entail the following:

- ✍ Separating the decision support system software into distributed components such as the decision support logic component that would contain the logic of the client's decision support software and the presentation component that will enable the client to use a rich graphical user interface to make various choices. These will be delivered to a storage component using XML, for storage into a database. The decision support logic component would then retrieve and process these using the client's proprietary algorithms to deliver the decision results back to the presentation component.
- ✍ All communication between these components will be via SOAP messages containing XML documents.
- ✍ The client could use the .NET My Services described above to deliver personal services to individuals. Individuals would use the MSN Passport to authenticate themselves and will then be given to access to the client's decision support systems to assist them with their personal decisions like car buying etc.
- ✍ The client's Corporate website is currently being converted from a HTML based architecture to an XML based format, based on the client's proprietary XML specification, which is also an XML-based language. This will enable pages of the client's corporate website to be made seamlessly available to the decision support system's GUI components passing the data back and forth using XML documents. The client's corporate website can actually be a component of the entire the decision support system under the .NET Framework.

8.2 An Analysis of Sun's J2EE Architecture

8.2.1 Description of J2EE Architecture

The Java2 Platform, Enterprise Edition (J2EE™) technology provides a component-based approach to the design, development, assembly, and deployment of enterprise applications. The J2EE platform offers a multi-tiered distributed application model, the ability to reuse components, integrated Extensible Markup Language (XML)-based data interchange, a unified security model, and flexible transaction control. Not only can you deliver innovative customer solutions to market faster than ever, but also your platform-independent J2EE component-based solutions are not tied to the products and application programming interfaces (APIs) of any one vendor. Vendors and customers enjoy the freedom to choose the products and components that best meet their business and technological requirements.

The J2EE platform uses a multi-tiered distributed application model. Application logic is divided into components according to function, and the various application components that make up a J2EE application are installed on different machines depending on the tier in the multi-tiered J2EE environment to which the application component belongs.

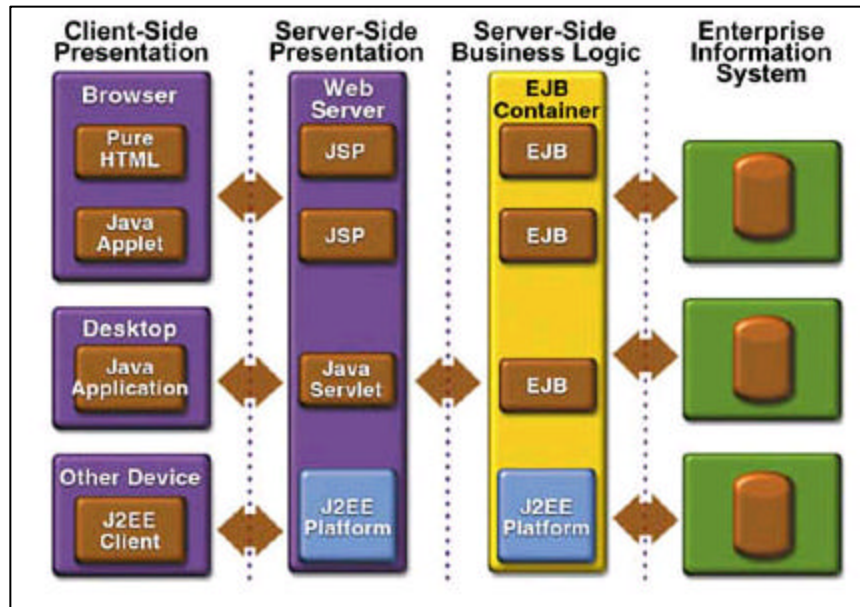


Figure 15: Standard J2EE Architecture

Figure 16 shows two multi-tiered J2EE applications divided into the tiers described in the following list. Client-tier components run on the client machine. Web-tier components run on the J2EE server. Business-tier components run on the J2EE server. Enterprise information system (EIS)-tier software runs on the EIS server.

Although a J2EE application can consist of the three or four tiers shown in Figure 16, J2EE multi-tiered applications are generally considered to be three-tiered applications because they are distributed over three different locations: client machines, the J2EE server machine, and the database or legacy machines at the back end. Three-tiered applications that run in this way extend the standard two-tiered client and server model by placing a multithreaded application server between the client application and back-end storage.

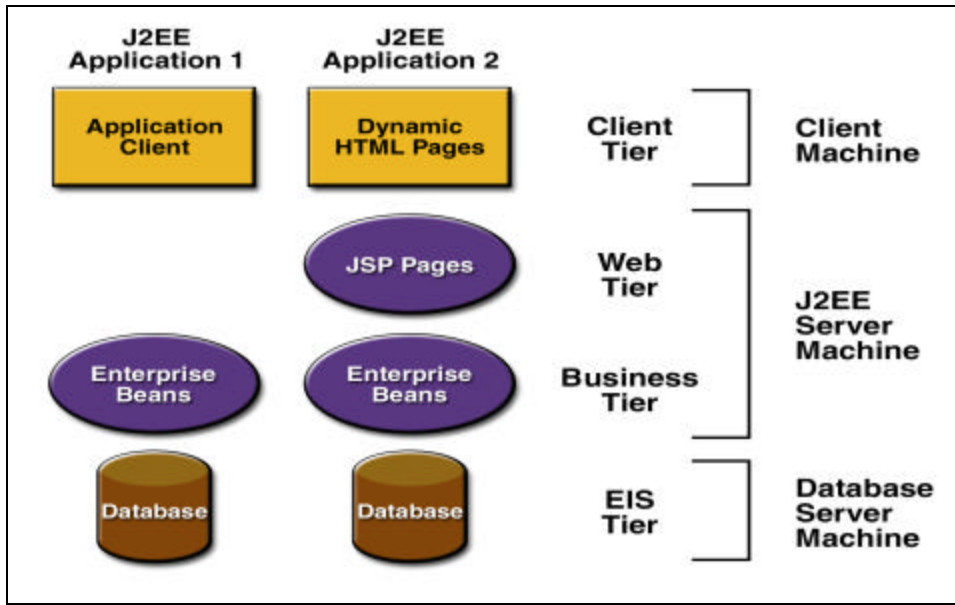


Figure 16 Multi-tiered Applications

The Java 2 Platform, Enterprise Edition (J2EE) technology provides a component-based approach to the design, development, assembly, and deployment of enterprise applications. The J2EE platform offers a multi-tiered distributed application model, the ability to reuse components, integrated Extensible Markup Language (XML)-based data interchange, a unified security model, and flexible transaction control. Not only can you deliver innovative customer solutions to market faster than ever, but your platform-independent J2EE component-based solutions are not tied to the products and application programming interfaces (APIs) of any one vendor. Vendors and customers enjoy the freedom to choose the products and components that best meet their business and technological requirements.

The J2EE specification defines the following J2EE components:

- /// Application clients and applets are components that run on the client.
- /// Java Servlet and JavaServer Pages (JSP) technology components are Web components that run on the server.
- /// Enterprise JavaBeans (EJB) components (enterprise beans) are business components that run on the server.
- /// J2EE Clients can be Web clients or application clients.

Figure 17 shows the various elements that can make up the client tier. The client communicates with the business tier running on the J2EE server either directly or, as in the case of a client running in a browser, by going through JSP pages or Servlets running in the Web tiers.

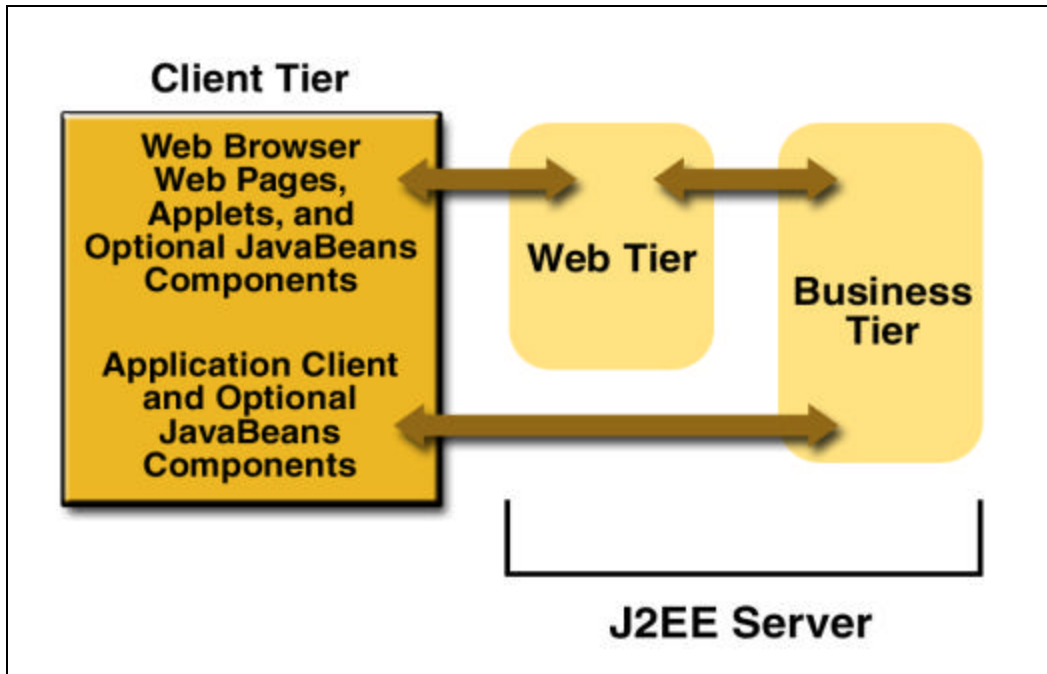


Figure 17: Server Communications

8.2.1.1 Web Components

J2EE Web components can be either Servlets or JSP pages. *Servlets* are Java programming language classes that dynamically process requests and construct responses. *JSP pages* are text-based documents that execute as Servlets but allow a more natural approach to creating static content.

Like the client tier and as shown in Figure 18, the Web tier might include a JavaBeans component to manage the user input and send that input to enterprise beans running in the business tier for processing.

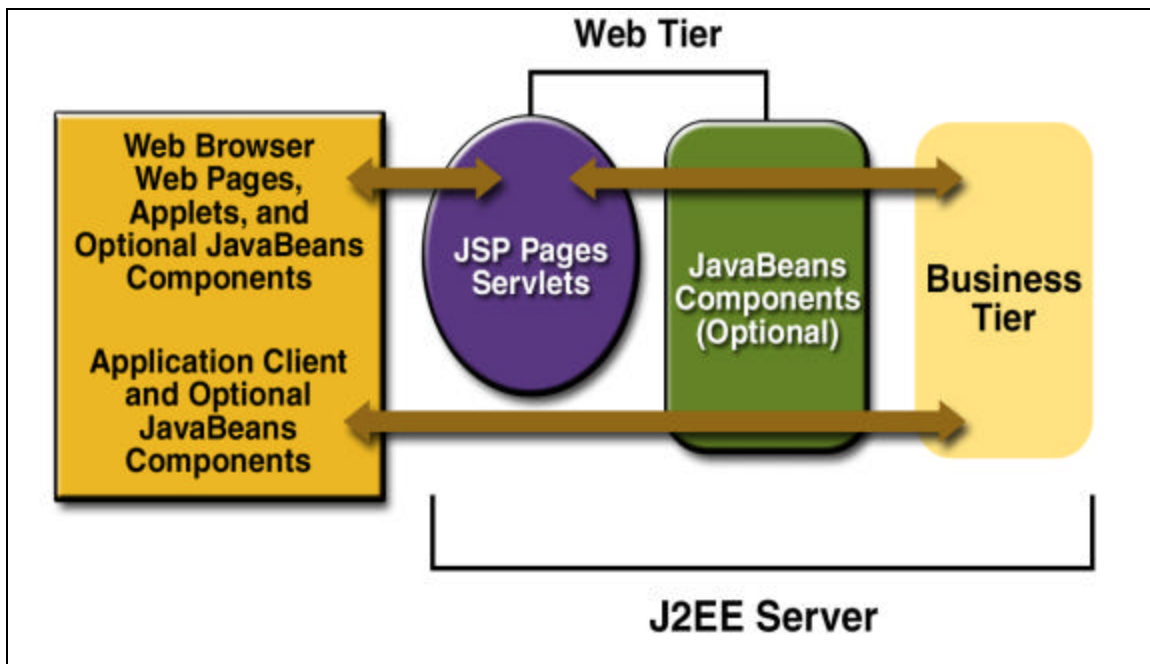


Figure 18: Web Tier and J2EE Application

8.2.1.2 Business Components

Business code, which is logic that solves or meets the needs of a particular business domain such as banking, retail, or finance, is handled by enterprise beans running in the business tier. Figure 1-4 shows how an enterprise bean receives data from client programs, processes it (if necessary), and sends it to the enterprise information system tier for storage. An enterprise bean also retrieves data from storage, processes it (if necessary), and sends it back to the client program.

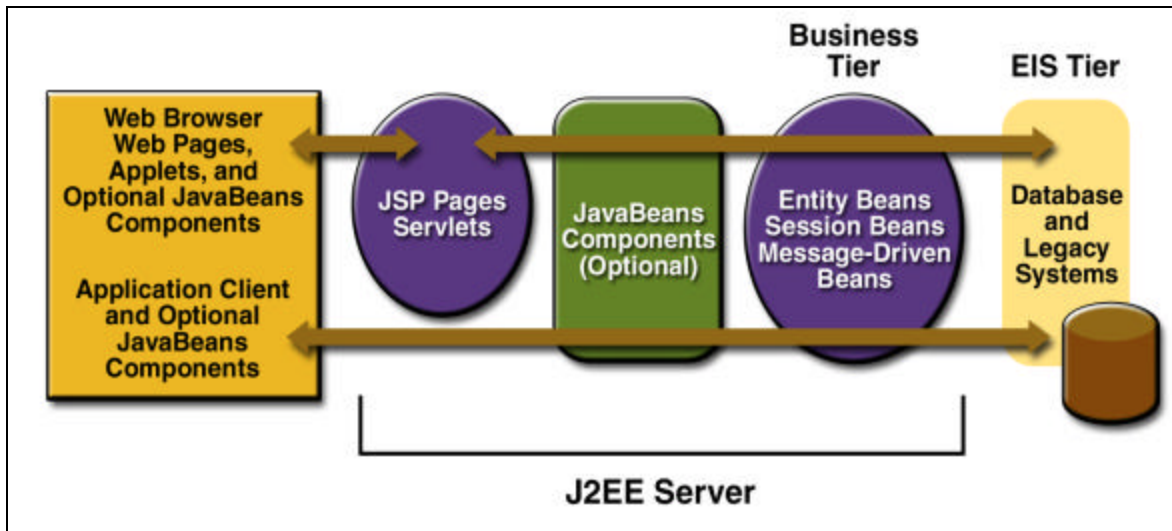


Figure 19 Business and EIS Tiers

8.3 Comparative Analysis of the Frameworks

J2EE and .Net have been evaluated against a set of criteria outlined below. These criteria have been described in detail below along with the pros and cons of each technology as pertaining to the specific criteria:

Time-to-Market: One way to speed time to market is to choose a platform that allows rapid application development. This enables developers to write and maintain code quickly, lowering development time. Both Sun J2EE and Microsoft .NET provide runtime mechanisms that insulate software developers from particular dependencies.

J2EE offers several features that accelerate time-to-market such as:

- State management services enabling developers to write less code and not worry about managing state, resulting in a higher degree of rapid application development.
- Persistence services (entity beans) enable developers to write applications without coding data access logic, resulting in leaner, database-independent applications that are easier to build and maintain. Programmatic transactions allow you to have greater transactional control. And custom tags are extremely powerful, and empower developers and web designers to easily collaborate.
- In addition to these features, there are several features that specific J2EE vendors offer, which aid time-to-market, such as business process management, E-Commerce components, XML-based legacy integration, and enhanced B2B collaboration.

Microsoft.NET offers a variety of time-to-market features:

- ASP.NET is independent of client device, and allows for user interfaces to be rendered to alternative user interfaces without rewriting code.
- Microsoft also offers Queued Components, which are superior to Message Driven Beans.
- .NET simplifies server-side programming by removing support for features found in traditional enterprise applications, such as stateful servers and simple transactions.
- Microsoft also provides business process management and E-Commerce capabilities, which are available in some J2EE implementations but not all.

In conclusion, the ability to achieve rapid application development offered by both J2EE and .NET is comparable. The feature differences are minor and it is difficult to make a compelling argument either way. Organizations should make their platform decision based upon larger business issues at hand that will ultimately dictate the platform choice.

Support for Existing Systems Most large corporations have existing code written in a variety of languages, and have a number of legacy systems, such as CICS/COBOL, C++, SAP R/3, and Siebel. Businesses normally do not have the funds or the time to reinvent all existing systems and hence corporations are always looking for rapid and efficient means to preserve and reuse their previous development efforts. Thus legacy integration often is one of the most challenging tasks to overcome when building enterprise wide applications or a web service

J2EE offers legacy integration in the following ways:

The Java Message Service (JMS) integrates with existing messaging systems and web services to integrate with any system. CORBA for interfacing with code written in other languages that may exist on remote machines. JNDI for loading native libraries and calling them locally are also available.

The most important part of the J2EE vision for integration is the J2EE Connector Architecture (JCA). The JCA is a specification for plugging in *resource adapters* that understand how to communicate with existing systems, such as SAP R/3, CICS/COBOL, and Siebel. These adapters are reusable in any container that supports the JCA. The major vendors of existing systems are bought into the JCA. The JCA market is producing a marketplace of adapters that will greatly ease enterprise application integration. Integration with packaged applications and legacy systems will become much easier.

.NET offers Legacy Integration in the following ways:

Through the Host Integration Server 2000 COM Transaction Integrator (COM TI) can be used for collaborating transactions across mainframe systems. Microsoft Message Queue (MSMQ) can integrate with legacy systems built using IBM MQSeries. BizTalk Server 2000 can be used to integrate with systems based on B2B protocols, such as Electronic Data Interchange (EDI) (the reader should note, however, that BizTalk does not serve as an access point to a proprietary network on which EDI takes place).

Maturity of Platform: Organizations that adopt a distributed application platform must consider the maturity of the underlying platform. A less mature, earlier-generation platform is more prone to errors and problems. We came across differing opinions about the maturity of both these platforms. Some consider J2EE to be a more mature platform quoting examples of a number of existing J2EE deployments that are running a variety of mission-critical business problems today.

On the other hand some experts say that .NET is the more mature platform basing their arguments on the fact that the first of the J2EE specifications, the EJB specification,

came out in 1998, three years after the first implementation of the equivalent .NET platform technologies, MTS, the forerunner of COM+. They also site examples of high volume and highly reliable web sites (NASDAQ and Dell being among many examples) that are using .NET technologies. However, it cannot be denied that both the platforms have some identifiable areas of risk associated with them.

✍ **Language Support:** J2EE is a specification that supports only Java-centric computing. Therefore all components deployed into a J2EE deployment (such as EJB components and Servlets) must be written in the Java language. On the other hand the .NET platform supports every language except Java (although it does support a language that is syntactically and functionally equivalent to Java, C#). Considering the importance of .NET, it also seems likely that any language that comes out in the near future will include support for the .NET platform.

The multiple language support that Microsoft has introduced with the CLR is an exciting innovation for businesses. It is clearly a feature advantage that .NET has over J2EE. However there are many arguments that go in favor of systems that are built using a single language. Some of there are:

✍ **Maintainability:** A combination of languages running in the CLR may lead to a mess of combination spaghetti code that is very difficult to maintain. An application written in multiple languages would require experts in different languages to debug and maintain it, which would lead to an increase in developer training expenditures.

✍ **Knowledge building:** Developers are unable to share best practices when working with combination language code. There are also arguments against both .NET and J2EE for situations when companies are considering integrating their legacy systems built using languages like COBOL. One way for organizations to make legacy code interoperable with Java is to do it through CORBA, which according to some is not a good strategy because they consider it to be a dead architecture.

Similarly for .NET even though it is claimed that through the use of CLR, legacy code can be made to seamlessly integrate into .NET it does not seem possible because languages like COBOL or VB were never intended to be object-oriented. Thus it is easy to understand that there are pros and cons to both approaches to language support and it makes sense for organizations to make their judgment based on what best suits their business needs.

✍ **Migration from Previous Platform:** Migration for organizations that have an existing deployment using J2EE-based technologies is not a big problem. Even though the Java Connector Architecture (JCA) and the web services support in J2EE is require new code, these are not considered as big changes to the existing code. There are certain apprehensions about the migration paths from VB or COM+ to .NET. This is because, to accommodate a Common Type System (CTS) of .NET, which standardizes on data types used between languages, the original Visual Basic data types have been dismissed. This might lead to a situation where code dependent upon those original Visual Basic data types might break. Also, to be able to take advantage of the managed code feature of CLR, developers will have to rewrite the exiting code as CLR code. These arguments suggest that the migration path will to .NET might not be as easy as the migration path to J2EE.

✍ **Portability:** A key difference between J2EE and .NET is that J2EE is platform-agnostic. It can run on a variety of hardware and operating systems, such as Win32, UNIX, and Mainframe systems. This portability is possible because the Java Runtime Environment (JRE), on which J2EE is based, is available on any platform.

J2EE is a standard, and so it supports a variety of implementations, such as BEA, IBM, and Sun. The danger in an open standard such as J2EE is that if vendors are not held strictly to the standard, application portability is sacrificed. To help with the situation, Sun has built a J2EE compatibility test suite, which ensures that J2EE platforms comply with the standards. This test suite is critical because it ensures portability of applications. By way of comparison, .NET only runs on Windows, its supported hardware, and the .NET environment.

However the choice of platform even with respect to portability is somewhat of a business decision. If a firm is selling software to other businesses, or if its a consulting company, having customers that are using a variety of platforms then it makes more sense for them to go in for the J2EE architecture.

Tools Some of the tools and Integrated Development Environment's available for J2EE are:

- o Forte: a modular and extensible Java-based IDE that pre-dates both Sun J2EE and .NET.
- o Web Gain's Visual Café,
- o IBM's VisualAge for Java,
- o Borland's Jbuilder.

Microsoft has always been a strong tools vendor, and that has not changed. As part of its launch of .NET, Microsoft released a beta version of the Visual Studio.NET integrated development environment. Visual Studio.NET supports all languages supported by earlier releases of Visual Studio. It also supports C#, Microsoft's new object-oriented programming language.

Even though the functionality of the toolset provided by J2EE vendors is considered to be better than the functionality of the tools provided by Microsoft, these tools are not 100% interoperable, because they do not originate from a single vendor. Microsoft on the other hand provides tools that are easy to use, and are considered to be very good to have when building web services. Our conclusion is that Microsoft has the clear win when it comes to tools.

System Cost: Both Microsoft and J2EE provide solutions that allow companies to choose your service level. For example, with J2EE it is possible to choose a high-end, expensive solution (iPlanet running on Sun Solaris in an E-10000 server), or a low-end, inexpensive solution (jBoss running on Linux on a Cobalt RAQ server). There are also an assortment of free and/or open source tools and services that support Java and XML.

.NET also provides a good choice in the selection of the system even though all the products available in this space are all Microsoft products. However it is important to realize that the price of the platform is only a small proportion of the total cost of implementing projects using any of these architectures. These can be is defined as the price of the server platform, the cost to train developers, the cost to build and evolve a solution on that platform, the cost to maintain the solution, and any business opportunity costs from picking the 'wrong' platform.

Therefore, it is important to realize that the total cost of ownership of a project is much more significant than any short-term cost differences between the underlying platforms. Therefore the decision to adopt any one of these cannot solely be based on the cost of just the platform.

Performance: A platform performs if it yields an acceptable response time under a specified user load. However, the definition of what is 'acceptable' changes for each business problem. The amount of traffic on the Internet today requires companies to build

the underlying infrastructure in a manner that it empowers them to build high-performing systems capable of handling that user load.

The primary bottleneck when building applications is usually integration with back-end database systems. The reason for this is that most enterprise applications are data-driven systems with much more data logic than business logic.

J2EE reduces database traffic through two tactics:

Stateful business processes: These allow organizations to maintain business process state in memory, rather than writing that state out to the database on each request.

Long-term caching (provided by some implementations) allows for database data to be cached for long periods of time, rather than re-reading database data upon each request.

It would be important to mention here that both maintaining business state in-memory and caching must be used with caution, and may result in problems if developers are not properly trained to use these features. This is a fundamental difference between the J2EE and .NET approaches to building applications: One advantage that J2EE provides is that it gives programmers more control over lower-level services such as state management and caching. Therefore experienced developers can use these features to improve the quality of their deployment. But it is of vital importance that developers are properly educated on when to make these tradeoff decisions, or error may be introduced into systems.

Microsoft.NET does not offer these tactics for improving performance but at the same time, there are no opportunities for developers to introduce errors into systems. Therefore, as is the case with most other parameters that we have used to compare these two platforms, for performance also both these alternatives have their plus and minus points. Thus the decision to adopt one out of these two alternatives would need to be based on not just the platform performance in isolation but also the quality of developers that are available with the organization.

Scalability: Scalability refers to the ability to add more workload. In web-based systems typically the additional workload comes through the addition of clients. Therefore a scalable system is one, which can handle peak user traffic without a drop in its response time. Scalability is essential when growing a web services deployment over time, because one can never predict how new business goals might impact user traffic.

A platform is scalable if an increase in hardware resources results in a corresponding linear increase in supported user load while maintaining the same response time. By this definition, the underlying hardware (Win32, UNIX, or Mainframe) is irrelevant when it comes to scalability, because both J2EE and .NET allow one to add additional machines to increase user load while maintaining the same response time. The major implementations based on J2EE architecture, as well as .NET, provide load-balancing technology that enables a cluster of machines to collaborate and service user load that scales over time.

The significant difference between J2EE and .NET scalability is that since .NET supports Win32 only, a greater number of machines are needed than a comparable J2EE deployment due to processor limitations. This multitude of machines may be difficult for organizations to maintain.

Recommendation:

Based on a critical evaluation of the two platforms, we recommend the J2EE architecture for the client's decision support software. Some of the critical factors in favor of J2EE are: This framework has been around for a longer time, and there is a large installed base of applications

on this framework. Skills for developing applications are easily available since this is JAVA based. Technically, one J2EE compliant system can be ported to another J2EE platform. This becomes important because the client could then deploy the decision support software on their customers' systems.

Scalability and performance is not an issue with the J2EE platform. The developer can focus on the 'business logic'. Transaction management, persistence, database connections, garbage collection, security is managed by the J2EE server.

We believe that porting the client's decision support software to the J2EE architecture will make a fundamental difference to how the client's decision support software is technologically deployed and delivered, and also holds the potential to make a difference to its business model and revenue streams.

9 Migration plan to J2EE

9.1 Description of J2EE Architecture

A detailed description of the J2EE architecture is given in the earlier section of the report. In this section, we will analyze the components of the J2EE architecture. A high level diagram of the J2EE architecture is depicted in Figure 20 below:

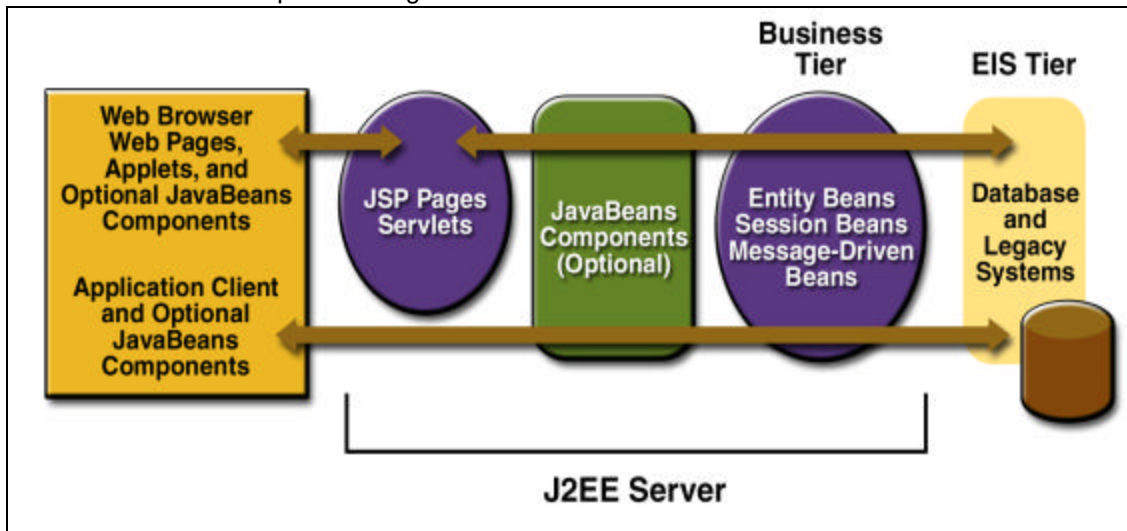


Figure 20: The J2EE architecture

In the three-tier architecture shown in Figure 20, the presentation logic resides on the client (first tier), the business logic resides on the middle tier (second tier), and other resources like the database reside on the backend (third tier). All distributed object protocols are built on the same basic architecture, which is designed to make an object on one computer look like is residing on a different computer. Distributed object architectures are based on a network communication layer that is really very simple.

The J2EE server is the heart of the architecture. The server powers the software components that drive the application and business logic. The Enterprise Java Beans (EJB) forms the core of the business logic. While developing applications on this platform, all the business logic is embedded in the EJBs. The developers are largely responsible for designing the business process flows and the business logic while the server manages other requirements like security, transaction management and persistence. EJB is a component model for component transaction monitors,

which are based on distributed object technologies. For the purpose of this paper, we will also refer to this as the business object.

The EJBs come in three fundamentally different types: entity, session and message driven beans. Both session and entity beans are RMI based server side components that are accessed using distributed object protocols. The message driven bean is an asynchronous server side component that responds to JMS synchronous messages.

Entity beans model real world objects. These objects are usually persistent records in some kind of database. For example, an entity bean might represent a customer, a piece of equipment, an item in inventory, amongst others.

Session beans are an extension of the client application and are responsible for managing processes or tasks. A session bean provides methods for doing things directly to an object, but doesn't say anything about the context under which those actions need to be taken. A session bean may need to make use of different entity beans to do a particular assigned task.

Session beans can either be stateful or stateless. *Stateful session beans* maintain conversational state when used by a client. Conversational state is not written to a database. It's the state that is kept in memory while a client uses a session. Maintaining a conversational state allows a client to carry on a conversation with an enterprise bean. As each method on the enterprise bean is invoked, the state of the session bean may change and that change may affect subsequent method calls.

Stateless session beans do not maintain any conversational state. Each method is completely independent and uses only data passed in its parameters. These kinds of beans do not need to maintain any conversational state from one method invocation to the next. Stateless session beans provide higher performance in terms of throughput and resource consumption than entity and stateful session beans because only a few stateless session bean instances are needed to serve a number of clients.

Message driven beans are integration points for other applications interested in working with EJB applications. Java applications or legacy systems that need to access EJB applications can send messages via JMS to message driven beans. The message driven beans can then process those messages and perform the required tasks using other entity and session beans. Message beans, in many ways, fulfill the same role as session beans by managing the workflow of entity and session beans to complete a given task. The task to be completed is initiated by an asynchronous message sent by an application using JMS. While session beans respond to business methods invoked on their component interfaces, a message driven bean responds to asynchronous messages delivered to the bean. Since the messages are asynchronous, the client that sends them doesn't expect or wait for a reply. The messaging client simply sends the message and forgets about it.

9.2 Current Architecture of the Decision Support System

The current architecture of the decision support software is as depicted in the figure 21 below.

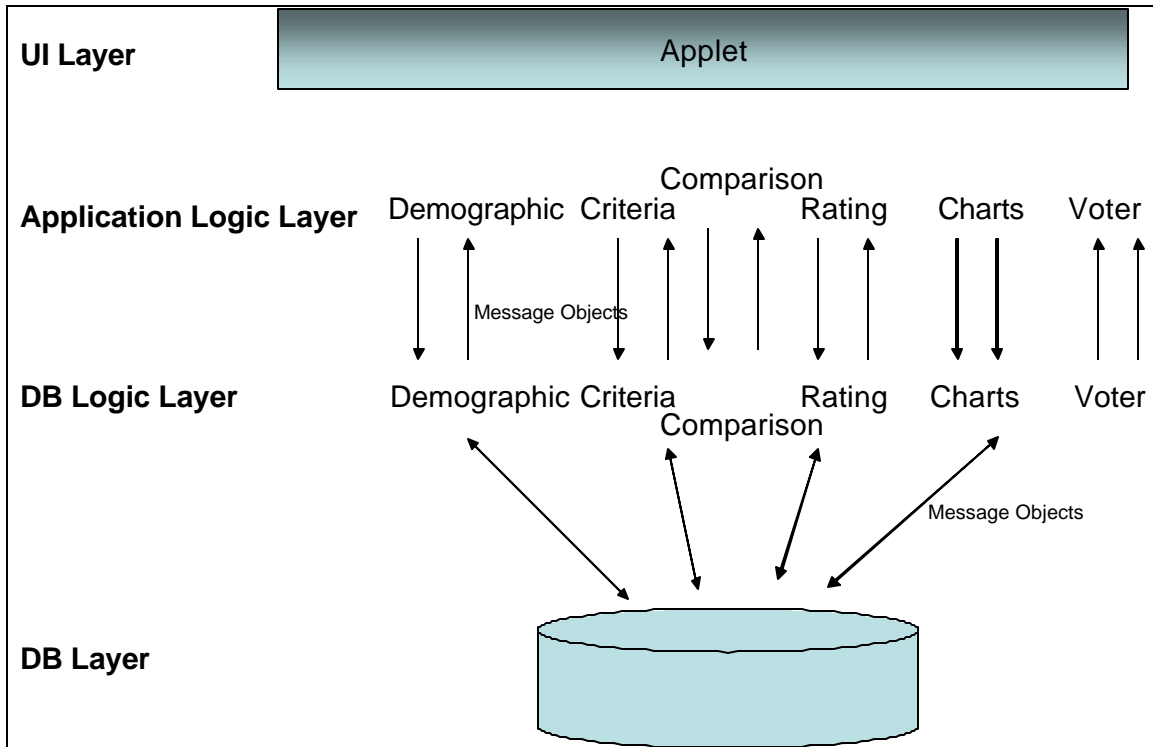


Figure 21: Current Architecture

The UI Layer is a desktop application-based Graphical user interface that guides the user through the decision making process. The application logic layer has a model that is composed of a set of items like:

9.2.1.1 Demographic questions

This comprises a set of questions a set of questions and a set of similar answers. This would be information like user information like the user name, password, email id, question 1/2, answer 1/2 and user id.

9.2.1.2 Criteria

This comprises a set of criteria using which the decision in question would be evaluated. For corporate decisions these could be criteria like: growth, increase in sales, profits, market share, amongst others. For personal decision making, like purchasing a car, these could be criteria like aesthetics, cost, brand, salvage value, maintenance cost, and fuel mileage, amongst others.

9.2.1.3 Comparisons

These are the comparisons of one criterion over the other in the analytical hierarchy process (AHP). This is used in assigning the global and the local weights.

9.2.1.4 Ratings

These are the user inputs for the rating of the criteria in question.

9.2.1.5 Charts

Based on the user inputs on the comparison of criteria, charts are generated by the system.

The communication between the objects occurs via the message objects. These message objects are required to communicate between the different layers. One of the concerns of the client has been the proliferation of the message objects during the process. The current configuration of the system does not enable an efficient garbage collection mechanism leading to this issue.

9.3 Proposed J2EE architecture for the Decision Support System

Based on a detailed analysis for the client's requirement, the team proposes the following J2EE architecture for the client's decision support software. Some of the key requirements for the system were that it should be:

- /// Web ready
- /// Web Services ready
- /// P2P ready

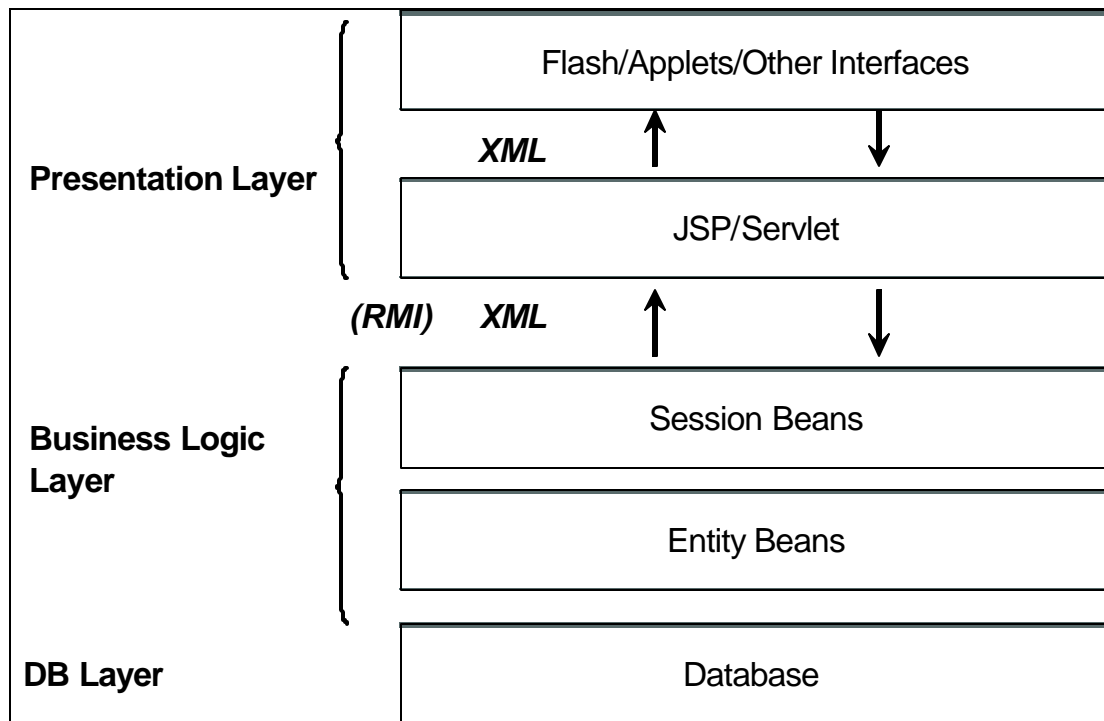


Figure 22: Proposed architecture of the system

9.3.1 Important Features of the architecture

The proposed architecture is the standard J2EE architecture with modifications to meet the requirements of the architecture being web ready and P2P ready. Here, P2P refers to the ability of clients to know where their peers are in the decision making process and does not refer to the traditional definition of P2P, i.e. the ability to operate without a central server. When the group session is in progress, the participants should be able to see the inputs entered by any user. In the current project, we have implemented by using a database polling mechanism, wherein we poll the database on a periodic interval to determine the state of all inputs.

Even with a J2EE framework, this kind of a requirement would not be met since J2EE does not have the feature of callbacks, wherein the Server can make functions / method calls on the client. It is for this reason, that we have recommended the Java Remote Method Invocation (RMI) architecture for callbacks between the presentation layer and the business logic layer. RMI will facilitate callbacks and when a user enters an input in the process, a callback will enable other

users to view this inputs. With time, this feature can be enhanced to allow users to have chat windows and exchange files, amongst other things.

In the earlier section of this project, we have discussed reasons for not selecting JSP/Servlet as the sole technology. We continue with the same argument for development of this architecture. JSP/Servlet is required to bind the presentation layer with the business logic layer, and to implement application logic.

Using XML rather than parameter passing as the mode of data transfer between the GUI (Presentation Layer) and the JSP/Servlet (Application layer) makes the presentation layer independent of the application layer. Therefore it is possible to seamlessly integrate graphical user interfaces of various kinds (Desktop GUIs, Flash, Applet, SVG) with the presentation layer with minimal rewriting of the application logic code.

Similarly, the application layer and the business layer have been made independent of one another since the mode of data transfer between these layers is in XML. This enables the architecture to be much more flexible.

9.3.2 GUI

There will be a number of GUIs that will be required for user interaction at different levels:

The client's decision support software enables decision making in three different ways:

- /// Common Comparison
- /// Data Comparison
- /// Matrix Comparison

Each of this decision making process has a different user interface. The system will need to have GUIs for each of these decision making process.

Users are also required to register by providing their details as explained in the earlier section under "demographic details".

After all user inputs have been entered, the system calculates the outcomes and presents the results to the user. This is usually presented as charts.

The system allows user to do a 'sensitivity analysis' based on changing the user inputs and analyzing the impact of their actions on the output.

For corporate decision-making, the system requires a one-time set up of the decision making process. This contains information like the project details, the voter details, and the criteria details. This will require a separate Admin GUI for this purpose.

9.3.3 JSP/Servlets

Corresponding to each of the GUI listed above; there will be JSP/Servlets that bind the GUI to the J2EE architecture.

9.3.4 Enterprise Java Beans

The team has identified in detail the respective stateful session beans, stateless session beans that contain the business logic. The entity beans that correspond to the database schema were also identified. Finally, the team identified the interactions between the session beans and the entity beans. This is part of the migration path that we presented to the client.

9.4 System Components:

In this section, we will discuss about the other components of the J2EE architecture.

While J2EE servers have been available in the market for a number of years now, databases are a relatively mature technology. In this section, we have done market research on the J2EE servers and the databases.

We have done a detailed study of the J2EE market, since there are a number of different servers available and there are distinct criteria for comparison. However, the study for databases is at a higher level since databases have more or less standard features.

9.4.1 J2EE Server

IBM's Websphere and BEA's Weblogic are the clear market leaders in this segment. The other players with smaller market shares are iPlanet, Oracle, Sybase and HP. The relative market shares of each of these players are depicted in the chart below:

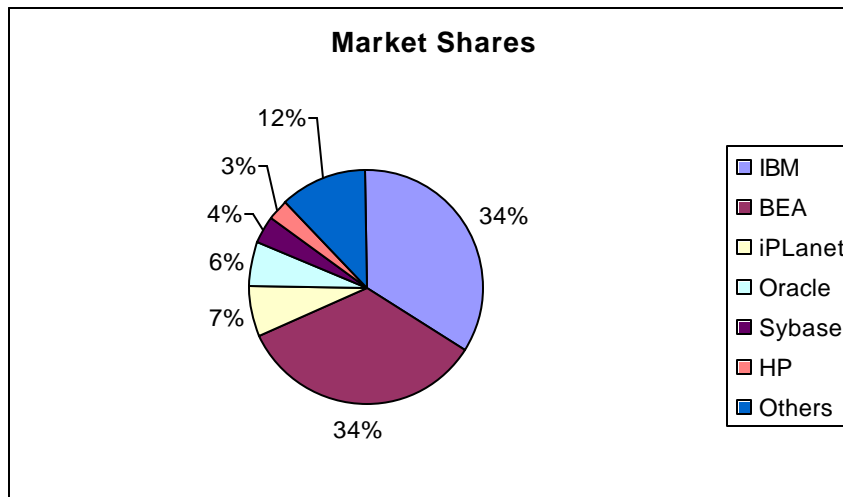


Figure 25: Market shares of the major players in the J2EE Server market

Source: <http://www.infoworld.com/articles/hn/xml/02/03/21/020321hngiga.xml>

The above information dated as of March 21, 2002. The team has done a survey of the J2EE server market for IBM, BEA, iPlanet, Oracle and Sybase. Information has been collected on the following parameters:

- /// Cost
- /// J2EE version supported
- /// Support for other applications
- /// Platforms supported
- /// Performance and scalability
- /// Security

A summary table for the products listed above is as shown below:

Product	J2EE Version	Supports	Platforms	Performance & Scalability	Security
Weblogic	1.2	JDK 1.3.1 EJB 1.1,2.0** JSP 1.2, JMS 1.0.2	NT, Solaris, HP-UX, AIX, Tru64, Win2K, OpenVMS, AIX 4.3.3,Sequent Dynix 4.4.4, OS/400 V4R4, Linux, SGI Irix 6.5,SNI Reliant 5.44C, Unisys OS1100, Unisys Burroughs, OS/390 V2R6	Supports the scalability, performance, and high availability essential to mission-critical e-commerce applications	Provides features like auditing, authentication, digital certificates, public key infrastructure, and single sign on
Websphere	1.2	JDK 1.3 EJB1.1 JSP1.1 JMS1.0.2+XA	NT, Win2K, Solaris, AIX, OS/400, HP - UX, Red Hat Linux, SuSE Linux, Turbo Linux, Linux/390, NetWare, OS/390	<p>WebSphere Application Server provides extensive performance, scaling, security, and control features for advanced e-business applications. New performance tuning wizards and improved resource and log analyzers help optimize transactional Web sites.</p> <p>A new Java Naming and Directory Interface (JNDI) cache stores name server requests on behalf of Web applications deployed in WebSphere. For applications that require many name server calls, application speed is substantially improved.</p> <p>With enhancements to the connection pooling algorithms and support for dynamic reloading of EJB</p>	WebSphere Application Server V4.0, Advanced Edition includes an interface for applications to interact with the WebSEAL component of Tivoli Policy Director. A Web application running in WebSphere calls the Policy Director product for coexistence with existing user and application policies, adding a level of authorizations for server-side execution environments. An upgrade of the Lightweight Directory Access Protocol (LDAP) client interface is also included for accessing directory services throughout the network. Added support for hardware crypto accelerators and smart cards to both Application Server and IBM HTTP Server helps improve the performance of protected client/server and server/server communications.

				components, WebSphere Application Server V4.0, Advanced Edition provides a highly scalable, high-performance deployment environment.	<p>Crypto hardware increases the server's throughput, lowering client wait time. Used with the storage feature, it provides secure management of private keys by storing them in dedicated hardware while in use and encrypting them when idle. Private keys never leave the module unencrypted.</p> <p>Smart cards enable greater mobility and enhance security by allowing users to carry their certificates with them.</p>
Oracle	1.2	JDK 1.2,1.3 EJB 1.1,2.0 JSP 1.1 JMS 1.0.2	Windows, Linux, Solaris, UNIX, Compaq Tru64, HP-UX	Oracle 9iAS is extremely lightweight, and runs, with a very small memory footprint, making it one of the fastest J2EE containers in use today.	Sufficient information not available at the moment.
Sybase EA Server	1.3	JDK 1.2.2, 1.3.1 EJB 2.0 JSP 1.2 JMS1.0.2	Windows, Solaris, HP-UX, AIX, Linux	Highly scalable, robust application server for e-portal and Internet business solutions.	Sufficient information not available at the moment.

Table: Summary table of J2EE Servers

9.4.2 Database

Oracle, IBM and Microsoft together have a market share of about 80% in the database market. Sybase, Informix and other small players put together have the rest of the market share. The relative market shares of each of these players are depicted in the chart below:

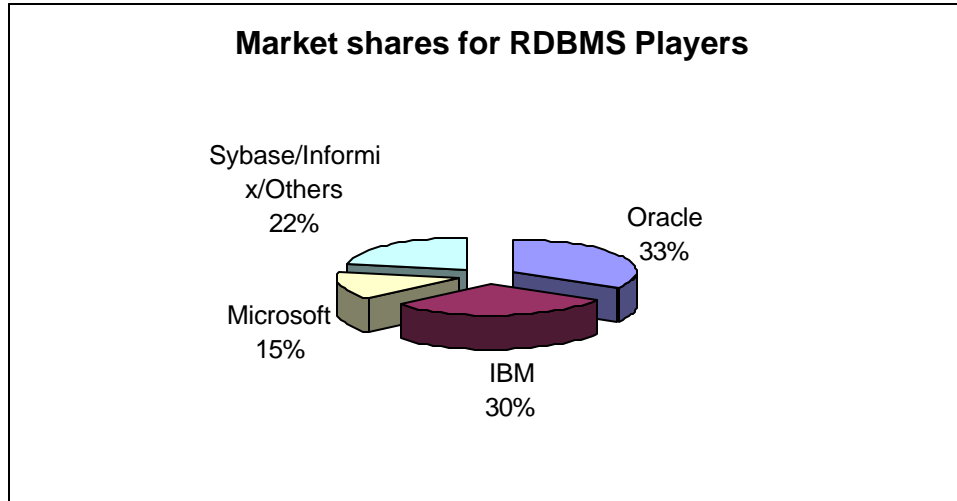


Figure 26: Market shares of the major players in the RDBMS market

Source: www.techweb.com

The above information is slightly dated, since it is for the 2001 market. Latest details of the market shares of different players and other market information can be purchased from Gartner.

Market shares can also be analyzed based on basis of the platforms. Microsoft is the fastest growing database on the Windows platform. As mentioned earlier, the performance of all the databases is more or less comparable. For databases, the criteria for evaluation usually are:

- /// The relational data model implemented by the product.
- /// Query handling and response time
- /// Connectivity and distribution
- /// Replication
- /// Integration with Internet applications

For large e-commerce websites, these issues are critical and would be the deciding criteria when deciding on the choice of a database. However, for the purpose of the client's requirements, these criteria might not be as critical while making the choice of the technology.

9.4.3 Recommendation

9.4.3.1 J2EE Server

Given the fact that the client currently has the Sybase EA server that has proven itself in the marketplace to be a good server, we recommend continue using the same server. In the short term, it might be a good idea to continue with the existing system configuration and re-deploy the application on the same platform.

In the long term, the company could think of migrating to either Websphere or Weblogic. This will, however, depend on the performance of Sybase, and the requirements of the company at a later point in time.

9.4.3.2 Database

Our recommendation for the database is also very similar to that of the J2EE server.

The client has the Sybase database that is a proven and an established product in the market. We recommend that the company uses the same database since we expect the database to perform to meet the client's requirements in the short term and in the long term. However, at a later date, if the client decides to migrate to the Weblogic or the Websphere server, integration

issues might need to be sorted out. However, given the open ODBC/JDBC standards, we don't expect this to be an issue.

9.4.4 Proposed implementation time line

We have identified the following key tasks that will enable the client to migrate to the J2EE platform.

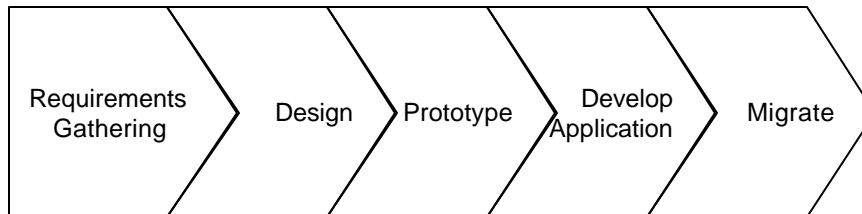
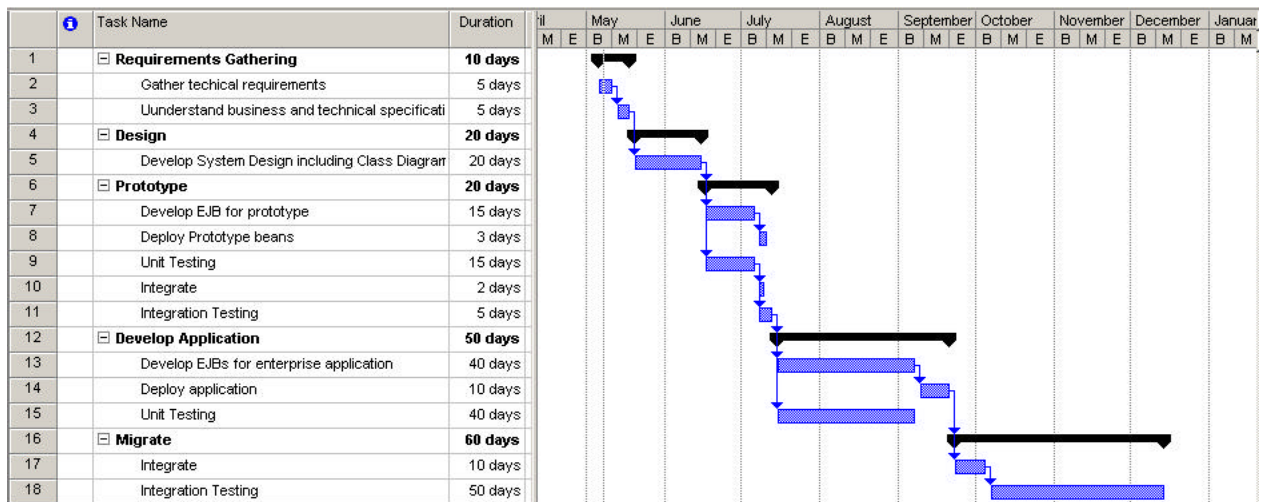


Figure 27: Proposed implementation timeline.

In this estimation, we have made the following assumptions:

- /// 6-member team will be working on the project
- /// The client's decision support software has 1 million lines of code
- /// 30% of this code comprises standard libraries
- /// Standard library code can be converted from C++ to JAVA @ 3000 lines of code per day.
- /// Other code can be converted from C++ to JAVA @ 1500 lines of code per day.

The proposed project plan for these activities is as shown below:



10 Marketing Plan

The team gained valuable insights about the direction in which the senior executives wanted to steer the company. These interactions have also given the team insights about the strengths of the decision making process and the value the client brings to the table.

We have looked at the following key components of the marketing plan for the client's company:

- /// Product and Service description
- /// Target Market
- /// Positioning
- /// Marketing of services

10.1 Product and Service Offered

The client's decision support software is a decision support tool that is based on the Analytical Hierarchy Process (AHP). This provides advanced decision-making products and technologies that enhance the focus and effectiveness of key corporate business processes.

The product enables the decision maker to come to a reasonable conclusion by taking the person through a structured decision making process.

The human brain goes through a complex decision making process. While this process is unstructured to a great extent, the brain follows a structure while processing the information. ***The decision making process guides the users through this and provides a 'structure' to the otherwise 'unstructured' analysis***

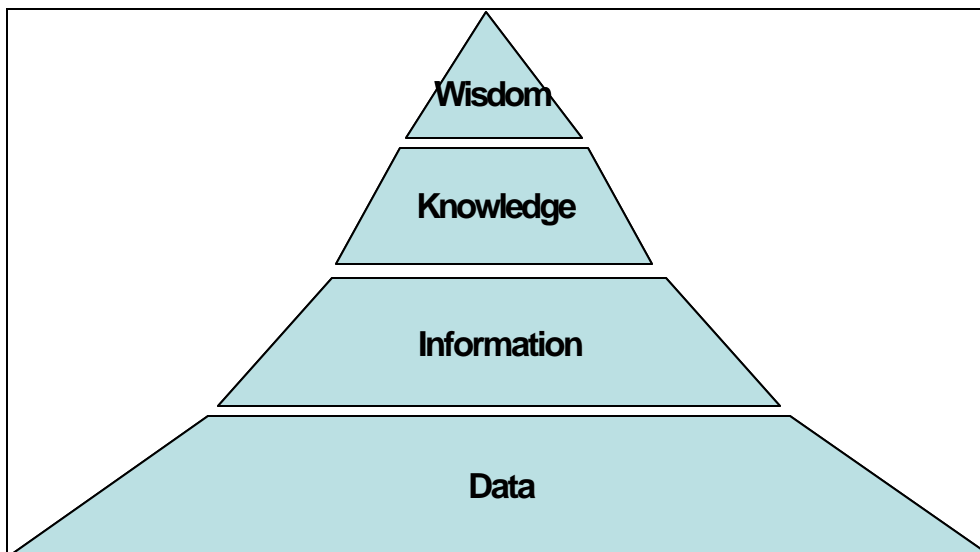


Figure 29: The knowledge hierarchy.

The product is a means of extracting the wisdom that resides in the minds of executives. In terms of knowledge management, this is the "intrinsic" knowledge that resides with the executives and is difficult to capture in any knowledge management systems.

The company could have enhanced product versions in the future. The decisions made by the company are stored in a database and become part of a rich repository. At a future date, companies can tap into this knowledge base to understand and learn what decisions were taken.

- ?? Target “big ticket” decisions.
- ✍ This product and the process will not be very valuable for routine, low cost decisions.
- ?? Avoid “retail” decisions.

As a starting point, the company could target the following target market segments:

✍ Automobile

- ?? Which automobile should the company introduce?
- ?? What automobile lines should the company put investment dollars into?
- ?? Which products should be phased out from the product portfolio?

✍ Pharmaceutical/Life sciences/ Healthcare

- ?? Which drug should be introduced next?
- ?? Which are the critical areas in which the company needs to focus and conducts research?

✍ Defense

- ?? Should the company bid for the next contract?

✍ High Technology

- ?? Which areas should the company direct their research dollars to?

10.3 Positioning

The positioning platform for the client’s decision support software could be:

- ?? A decision making tool that captures organizational knowledge.
- ?? Tapping the 'gut' feel or the intrinsic knowledge of managers is key in managing knowledge within enterprises. The process followed in the group decision process, under the guidance of a coach, taps this gut feel knowledge of the managers.

The client’s decision support software has a feature that enables customers to store past decisions. This will enable to analyze actions taken, things gone right, things gone wrong, amongst others. This will allow companies to convert the thoughts in the minds of the executives into quantifiable numbers along with their thoughts and comments. This would prove to be very beneficial tools for managing knowledge within the client enterprise.

Future versions of the client’s decision support software could have further enhanced features like the capability to index decisions, insert comments from executives, store files in different formats, amongst other things.

10.4 Marketing

The company should continue with the current direct sales model. More sustained marketing efforts should be made in the form of presentations to prospective client companies.

White papers on the technology and how it can benefit client companies could be published in industry publications and journals.

10.4.1 Partnerships

It is important for the client to have strategic alliances with other companies to be able to market their services more effectively. It will be important to have tie-ups with firms who are complementing the services of the client’s decision support software.

10.4.2 Potential Partners

There are a number of companies that offer knowledge management products and knowledge management services. Such a partnership would enable both companies to provide a “complete” solution to the customers.

Some potential partner companies could be:

- o Delphi Group (http://www.delphigroup.com/coverage/knowledge_management.htm)
- o Cogos (<http://www.cogos.com/>)
- o The Knowledge Company Inc. (www.theknowledgecompanyinc.com)
- o Cerebyte, Inc. (www.cerebyte.com)

Big 5 consulting companies:

Consulting companies usually provide companies with knowledge management consulting services. These companies have access to top management of their customers. Typical engagements with these consulting companies are strategy formulation, business process improvement, and business risk assessment amongst others. The advantage of such an alliance is that this is a complementary relationship, and could lead to access to top management of FORTUNE companies and related follow-on business. The disadvantage of such an alliance is that the client might be sidelined by these organizations when their own self-interests are not being met.

Besides, other potential partners could be other smaller consulting companies like: Answerthink Consulting (www.answerthink.com)

10.4.3 Pricing

Our recommendation in our Migration strategy document to migrate the client’s decision support software Architecture to a distributed J2EE based framework has important implications to the client’s pricing model.

Currently customers are charged based on a per-seat licensing model. When the client’s decision support software is implemented on the J2EE architecture, the client would be able to deploy the software in a number of ways

Deliver the entire application to the Customer, and the customer is free to distribute the application as he sees fit. A typical deployment would involve locating the brains of the decision support system and the storage at a central location, while deploying the presentation logic across the organization. This could be on similar terms as the per seat licensing fee pricing model being currently used by the client.

Host the brains of the system on the client’s decision support software servers while delivering the logic for presentation to the client. This would cost customers much lesser in terms of infrastructure by letting the client handle the centralized decision support system and storage. The client could build an ASP based pricing model for this wherein customers would pay a monthly fee for use of the client’s centralized decision support system.

Alternate scenarios could be hosting the decision support system on the client’s servers, outsourcing storage to Third parties and deploying presentation to Customers. The customer would then pay the client and the third-party host on a monthly basis.

Given the flexible distributed nature of the J2EE architecture, the client could thus build a number of licensing and pricing models for different deployment configurations.

11 Reflections / Conclusions

This has been a valuable learning opportunity for most members of this team for whom this had been a first experience with a real-life project. We as a team established a close working relationship with the client and the sponsor and we enjoyed the intellectual stimulation and learning that this experience brought.

We also experienced some of the fallout of a bad business cycle with our client wherein we took the fact that the client was having some problems with their marketing and business plans as a challenge and greatly extended the scope of our project to include a migration strategy for their Software as well as a pricing and positioning plan for the company, which involved a look at their current business model and recommendations on possible changes to their business focus.

One of the distinguishing features of this project from other projects was that we had a very clear sense of where our work fitted into the overall technology and business strategy of the company. We knew we could contribute in an attempt to make a difference to the company's fortunes.

A valuable lesson we learnt was that development of software does not take the usual "rules-of-thumb" principles of software engineering. In most real life projects you do not baseline your requirements analysis and go onto design. The requirements of the project do tend to evolve and change through the life of the project and attempts need to be made to ensure that the system is flexible enough to incorporate these changes. We had discussions with our client about the design and the behavior of components well towards the end of the implementation cycle. Software development also almost never follows the usual percentage split of activities. 20% on Analysis, 30% on Design, 20% on Coding and 30% on testing.

The unique features of each project will tend to shift the emphasis to one of these four development activities. In our case Coding took more of a share of the effort than either design or analysis because of the intensive coding effort needed on our part to build a front-end customizable application, one of our deliverables for the project.

In the end, it was a project that was as real-life as it gets and we had the opportunity to exercise the skill sets of all members of the team. We expanded the scope of our project to include a migration strategy and a business model/pricing and positioning strategy document. Therefore all members of the group had an opportunity to do what they liked, be it coding and development, higher level technical analysis and design or business analysis.

This was a good project experience and we believe that the future holds interesting times ahead for our client and we hope that our project has some positive impact to the operations of the company.