

On using Web technologies to architect DSS: The case of Support Requirements Planning

Ramayya Krishnan (rk2x@cmu.edu)

Rema Padman (rpadman@cmu.edu)

The Heinz School

Carnegie Mellon University

Abstract

Beginning with transparent and ubiquitous access to multimedia documents marked up in HTML, the Web has evolved (and continues to evolve) dramatically to incorporate a range of network computing technologies. These include network-centric languages such as Java, software component architectures such as ActiveX, and distributed object technologies such as CORBA. Paralleling these developments is the demand for more flexible and distributed access to decision support components (i.e., models, data and user interfaces), incorporation of rapidly changing information sources (e.g., streaming stock market data feeds), expanded use of notification services to disseminate results of analyses, and integration with related organizational applications (e.g., order placement) using messaging standards such as electronic data interchange (EDI). How should these recent developments in network-centric technologies be harnessed to meet these requirements? Using our work with Support Requirements Planning, a logistics problem encountered by military planners, to provide substantive context, we examine the alternative ways in which Web technologies can be used to architect decision support systems.

1. Introduction

The unprecedented growth of the World Wide Web has had a profound impact on the organizational use of information technology. The core technologies underlying the Web – the IP protocol and its accompanying TCP/IP protocol suite – have emerged as a standard way to create both intra-organizational and inter-organizational information networks. Distributed object technologies such as the Java object model (Arnold and Gosling, 1996), Common Object Request Broker Architecture (CORBA) (Siegel, 1996) and the Component Object Model (Chappell, 1996) are being integrated with the Web-based information networks to provide a ubiquitous, standards compliant computing platform. The Web browser has become the standard user interface to a wide range of information services and repositories.

The availability of such a network computing platform gives decision support system (DSS) developers and researchers the opportunity to revisit architectural choices made in an environment in which either proprietary desktop or local-area network architectures were dominant (Bhargava, Krishnan, Mueller, 1997a, 1997b, Juesfeld and Bui, 1995, Ba et al., 1995). Concurrent with these developments has been the evolution in end user requirements emphasizing the need for the sort of connectivity offered by information networks. These include platform independent access to DSS components (i.e., the unbundling of data, models and the interface), the need to incorporate data from remote, and possibly, streaming data sources (e.g., stock market data feeds), communicate customized reports based on DSS analysis to a distributed group of users, and integration with other organizational applications (e.g., order placement) using messaging standards such as electronic data interchange (EDI) (Hendry, 1993).

Given that several alternative architectures may be realized using Web-based technologies, how should these end user requirements for decision support be satisfied? What should be the criteria used to evaluate these alternatives and how do some important alternative architectures compare with one another? An analytic review of alternative Web-based architectures for DSS is the focus of the paper. To provide substantive context to our discussion, we use the information processing needs of Support Requirements planning, an important logistics problem encountered by military planners.

The rest of the paper is organized as follows. We begin in Section 2 with an introduction to Support Requirements Planning. We specify the problem precisely using a mathematical model and discuss the information processing requirements that arise in implementing decision support systems using the model. In Section 3, we provide a brief review of the conceptual model underlying basic web technology. In Section 4, we present a detailed discussion of the shortcomings of basic Web technology from a DSS perspective and discuss extensions to the basic technologies that address these problems. In Section 5, we present a set of metrics to evaluate alternative web-based DSS architectures. In Section 6, we discuss two stylized Web-based DSS architectures to meet the information processing requirements identified in Section 2, using the technologies presented in Sections 3 and 4. We present conclusions in Section 7.

2. Support Requirements Planning: An overview

A military force consists of a set of combat units (the units that fight) and a set of support units that sustain the combat force. Given a set of combat units that need to be deployed, the goal of support requirements planning is to identify the smallest set of combat support and service support units that need to be added to sustain the force. This is a complex task. At its core, there is the need to estimate demand for services (e.g., transport, maintenance of vehicles, medical care) imposed by combat forces and the capabilities of combat support and service support units to provide these services. Given these estimates, support requirements planning involves matching the supply capabilities of service support units with the demand requirements imposed by the combat units. In general, two objectives must be simultaneously satisfied during the planning process: a) the demand for services should be met, and b) the number of combat support and service support units added to the force to meet this demand should be as small as possible (Krishnan, 1995).

Traditionally, military planners have dealt with the requirements planning problem using detailed simulations of expected areas of conflict (e.g., the Korean Peninsula or Central Europe). However, with the end of the Cold War, there is the need to deploy forces at short notice into areas for which detailed plans do not exist (Reimer, 1994). In this *no plan* scenario, planners are

faced with the option of using their traditional approaches which rely on unvalidated assumptions (e.g., about terrain, weather, infrastructure) to generate estimates of demands and capabilities. An alternative which we consider in this paper is to use the reach offered by information networks to obtain data on a continuing basis from the field where an initial set of forces are deployed. This data from the remote site can be combined with data available at the planning site to plan in an *incremental manner*. The information flows that arise in this model of support requirements planning are illustrated in Figure 1.

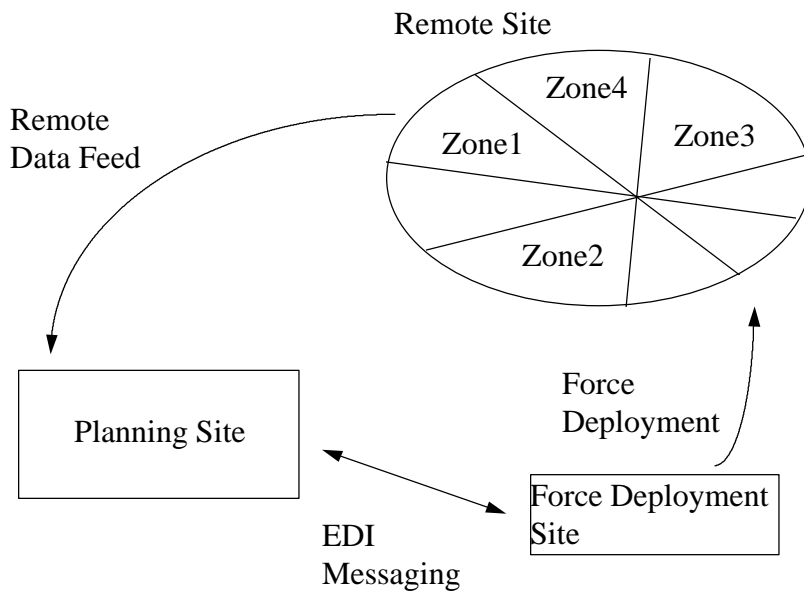


Figure 1: Support Requirements Planning

In concept, this is similar to the quick response and just in time approaches that have revolutionized inventory maintenance in the retail sector (Parker, 1996). A precise specification of the support requirements planning problem is necessary to understand the information processing requirements that arise in implementing decision support systems. We use a mathematical programming formulation (Bradley, Hax, Magnanti, 1977) to develop a specification of the problem.

2.1 The Model

Problem Statement: As stated above, the essence of support requirements planning is to match the capabilities of the support units with the demands for service imposed by combat units. Let *Demands* be the set of service types demanded by combat units. Let D_j be the amount of each service type j demanded. For example, if transportation is a service type, a demand for this service may be stated in terms of a requirement to move 2000 short tons per day. Combat support units (the set CS) and the combat service support units (the set CSS) have the capabilities to provide these services. Let C_{ij} denote the capability of a CS or CSS unit i to provide service j . For instance, a light transportation company, a combat support unit, can make four round trips per day and move a total of 1000 short tons. Given information about the capabilities of units and demands imposed by a force, the goal is to determine the smallest number of combat support and combat service support units that are required to meet the demand constraint.

The foregoing discussion assumes that the forces are all deployed in a single geographic zone. However, as illustrated in Figure 1, forces are deployed spatially into a set of geographic zones. Thus, demands for services arise in specific zones. Combat support and combat service support units have the capabilities to provide the required service in the zone into which they are deployed, and depending on conditions (e.g., terrain, hostility) on the field, to other zones as well. The ability to leverage economies from the capability of a unit to meet demands in zones other than the one in which it is deployed is the reason for working with a set of disaggregated zones.

Taking this spatial information into account leads to the mathematical formulation shown in Figure 2. The first constraint states the goal of meeting the demand imposed by a set of combat units for a set of services. The second constraint states the goal of having as few CS and CSS units as possible added to the force. The objective function associates penalties P_{jz} and Q with deviations from the first and second constraint sets, respectively.

$$\text{Min} \quad \sum_{j \in \text{Demands}} \sum_{z \in \text{Zones}} P_{jz} s_{jz}^- + Q d^+$$

s.t.

$$\sum_{i \in \text{CS} \cup \text{CSS}} \sum_{z_1 \in \text{Zones}} C_{ijz_1z_2} X_{iz_1} - s_{jz_2}^+ + s_{jz_2}^- = D_{jz_2} \quad \forall j \in \text{Demands},$$

$$\forall z_2 \in \text{Zones}$$

$$\sum_{i \in \text{CS} \cup \text{CSS}} \sum_{z \in \text{Zones}} X_{iz} - d^+ + d^- = 0$$

$$\sum_{z \in \text{Zones}} X_{iz} = 1 \quad \forall i \in \text{CS} \cup \text{CSS}$$

$$X_{iz} = 0 \text{ or } 1$$

X_{iz} is a binary variable which is designed to model the deployment of unit i in zone z . It takes on a value of 1 if the unit is deployed and takes on a value of 0 if the unit is not deployed.

$C_{ijz_1z_2}$ is a parameter which models the amount of demand type j that arises in zone z_2 which can be met by unit i in zone z_1 .

d_1^+ is the slack variable which models the number of support units that are added to the force.

d_1^- is the slack variable which plays the role of a dummy variable in this formulation.

$s_{jz_2}^+$ is the slack variable which models the deviation over and above required demand.

$s_{jz_2}^-$ is the slack variable which models the amount by which supply falls short of required demand.

Zones is a set of spatially distinct zones in which combat and support units may be deployed. It is assumed that the zone in which each combat unit is deployed is known.

CS is the set of combat support units from which units are selected.

CSS is the set of combat service support units from which units are selected.

Demands is the set consisting of the types of demands that are imposed by combat units.

P_{jz} is the penalty associated with deviations from the demand constraint.

Q is the penalty associated with deviations from the minimal forces constraint.

Figure 2: Mathematical Formulation of the Support Requirements Planning Problem

2.2 Information Processing Requirements

Data Requirements: The two data intensive model parameters are $C_{ijz_1z_2}$ and D_{jz} . $C_{ijz_1z_2}$ denotes the capability of a unit i in zone z_1 to provide service j in zone z_2 and D_{jz} denotes the demand for service j in zone z . These parameters map to the following relational schemata.

CapabilityTable(UnitType, Zone of Deployment, Zone of Service, Service Type, Capability)

An example record: 55500LC, z_1 , z_2 , AMPHIB-OFFLOAD, 2tons

DemandTable(Service Type, Zone of Service, Demand)

An example record: AMPHIB-OFFLOAD, z_1 , 5tons

While the set of CS and CSS units and demand service type data are available at the planning site, the capability and demand tables need to be obtained from the remote data source in the field. The user supplies the parameters P_{jz} and Q that specify the penalty weights on the deviations from the two constraint sets in the model. Thus, a combination of data from the user, data at the planning site and remote data feed from the deployment site is needed to instantiate the model. Of these data, the user may choose to modify the penalty weights or work with some subset of the data from the local and remote sites. An interesting feature of the data from the remote site is that it is dynamic. Over time, the demands for services as well as the capabilities of the units change due to conditions on the field. As modified data is retrieved from the remote site, this can trigger reinstantiation and execution of the model.

Algorithmic Processing Requirements: The model is an integer programming model and is NP-hard (Garey and Johnson, 1979). Solution of instances of the model of practical size will require the use of heuristic methods based on techniques such as Genetic Algorithms (Goldberg, 1989, Holland, 1975; Michalewicz, 1992) and Tabu Search (Glover, 1989,1990, Reeves, 1993). Another important requirement is the need for incremental planning. As the data from the remote site indicate changes in the capabilities and demands, the model will have to be solved again. This solution process should use as much of the knowledge gained from the previous planning runs. Thus, the heuristic methods should take advantage of the explicit need for incremental planning. A

related issue is the infrastructure required to trigger incremental planning when new data feeds are obtained from the field. This requires functionality similar to triggers (Elmasri and Navathe, 1994) in databases based on a type of notification service.

Notification Services: Three types of notification need to be supported.

- Notification of results

The first type is the communication of the results of the planning process at different levels of detail to a distributed group of users. This is a fairly typical requirement in most decision support applications. However, electronic access to these reports and to the models from which the reports were obtained are desired (Kimbrough et al., 1990).

- Inter-application messaging for EDI

The second type of notification relates to actions that are taken with respect to the support requirements plan. If the plan is approved, the decision support application should be able to communicate the plan to another organizational unit charged with acting on the plan using a messaging standard such as electronic data interchange (EDI) (Hendry, 1993).

- Notification for triggering incremental planning

The final type of notification is to enable coordination between the remote data source and the decision support engine at the planning site. This type of notification is also a form of inter-application messaging which is used to trigger incremental planning.

User Interface Requirements: The interface should, at a minimum, permit the user to

- request execution of the model explicitly or on the basis of notification that data from the remote site indicate replanning
- enable iterative interaction (e.g., what if analysis) with changes made to the model parameters during exploration of the decision space

- provide control over the choice of user supplied data, local data and remote data that is used to instantiate the model (e.g., enable the user to evaluate queries on a database to extract desired data)

Over and above these interface requirements related to supplying the data, the user might desire interaction with a direct manipulation interface (Jones, 1995) in order to interact with or to direct the process used to plan support requirements.

2.3 Summary

The requirements that have been discussed in the context of support requirements planning arise in other domains as well. As noted earlier, the information flows depicted in Figure 1 are widely observed in quick response and just in time inventory systems (Parker, 1996). We will discuss Web technologies before proceeding to examine Web-based architectures needed to satisfy the requirements identified in this section.

3. The World Wide Web: A Brief Overview

In this section, we briefly review the key technologies underlying the Web. Readers interested in a detailed discussion of Web technologies from a computational viewpoint are referred to a forthcoming article currently under preparation by Bhargava and Krishnan (1997).

3.1 The Conceptual Model

The Web is organized as a client server system (see Figure 3) (Berners-Lee et al., 1994; Yeager and Mcgrath, 1996). The Web browser is the client and generates requests for services to the Web server. Requests are made using the *Hyper text transfer protocol* (HTTP). HTTP requests are of two types. The first and most prevalent is the request for a specific document identified by its *Uniform Resource Locator* (URL). In response to such a request, the server retrieves and sends the requested document to the client. The server does not examine the contents of the

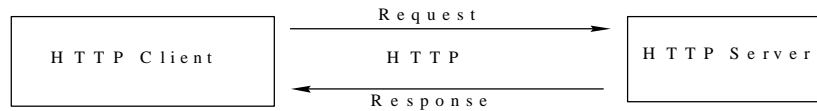


Figure 3: Client Server Architecture on the Web

document that is served. The browser processes the document and takes actions such as displaying the document. The second type of request is a request for computational services (see Figure 4). In response to this type of request, the Web server invokes an external program and transfers to it the data sent from the client to the server using a standardized application programmer interface (API) referred to as the *common gateway interface (CGI)*. Results of the computation are transferred back to the server and onward to the Web browser. In this manner, documents can be created on demand based on the results of external computations.



Figure 4: The Common Gateway Interface

The client server model underlying the Web has key implications for the way in which each of the three principal components of the DSS can be unbundled and provisioned. For example, the user interface to the DSS can be made available through a universal client (i.e., the Web browser). The data and the models can be made available as external applications invoked by a Web server. While this sort of unbundling was in principle possible with Local Area Network-based client

server systems (Orfali and Harkey, 1994), the ubiquitous and standards compliant nature of the Web promises to extend the reach of DSS over a wide area network.

3.2 Hypertext Markup Language (HTML)

HTML is the standard language used to mark up documents on the Web. It is a simple language and is a derivative (more specifically, a document type definition (DTD)) of the more powerful Standard Generalized Markup Language (SGML) (Van Herwijnen, 1994). All Web browsers are capable of interpreting and displaying HTML. Thus any HTML document on the Web is universally readable. The simplicity of HTML also enables individuals to easily author content on the Web using HTML. Finally, HTML forms permit data to be sent to the Web server from the browser. This makes the Web a two-way medium and is the feature used to communicate user-supplied data to CGI-based applications.

4. A DSS Centric Analysis of Basic Web Technology

The original model of the Web that was reviewed above has some major shortcomings with respect to the requirements of decision support systems. These are discussed below along with the extensions that have been introduced to deal with these problems.

4.1 User Interface Limitations

HTML is a simple and effective language for marking up hypermedia documents. However, it does not have the capability to describe direct manipulation interfaces. To the extent that the decision support application requires such an interface, HTML capabilities will have to be extended.

- **Applets**

Applets are small applications written in programming languages such as Java (Arnold and Gosling, 1996). Applets are transported using HTTP to the client using HTML as a container. An

illustrative fragment of HTML code developed by Chris Jones at the University of Washington (<http://weber.u.washington.edu/~cvj/download.html>) to serve as a container for an applet which implements a direct manipulation interface for linear programming is shown below.

```
<APPLET codebase="Beta/Classes" code="NULP.class" width=600 height= 500>
<PARAM name=objsense value="max">
<PARAM name=objective value="6 5">
<PARAM name=constraints value="1 2 < 18 2 1 < 18 1 1 > 3 1 -1 < 6 -1 1 < 6">
<PARAM name=graphwidth value=200>
<PARAM name=graphheight value=200>
<PARAM name=showrect value="-1 -1 15 15">
<PARAM name=varlabels value="Ch Ta">
<PARAM name=roundrhsto value=0.1>
</APPLET>
```

The applet referred to in the *codebase* field is loaded using HTTP, and parameters to its methods are specified using the PARAM tag. Applets can implement arbitrarily complex interfaces. Given the widespread availability of Web browsers with Java interpreters, Java is the language of choice to implement complex user interfaces.

4.2 Transport Protocol Limitations

Hypertext transfer protocol (HTTP) is the transport protocol used on the Web. It is a connection-oriented and stateless protocol. The connection oriented nature of HTTP refers to the TCP/IP connection being held open by the Web server as long as it takes to service a request. When external computation is required to service a request from a Web client, as is the case with decision support applications, this might result in connections being held open for long periods of time. This can result in inefficient use of network resources.

The stateless nature of the protocol implies that there is no relationship between adjacent HTTP requests. This makes sense in the hypermedia context where each jump following a URL might be to a different server. In the case of decision support applications with its emphasis on *what if* analysis and exploration of the decision space, there is the need to maintain stateful interactions between Web clients and Web servers.

- **Addressing the stateless nature of HTTP**

In response to this problem, several proprietary extensions have been made either to the protocol or to the common gateway interface specification. For instance, Netscape servers and browsers process an additional field in the HTTP header referred to as the *cookie* (Yeager and Mcgrath, 1996). The cookies permit a server to recognize the browser from which a series of requests are being generated. This enables the server to link the information being supplied across requests. Alternatively, scripting programs written in languages such as Javascript on the web server provide similar type of support for stateful interaction with an external program. This is an important feature for network-centric decision support systems since, in its absence, one would have to supply the entire model instance as part of each request for sensitivity or what-if analysis.

- **Addressing the connection-oriented nature of HTTP**

When requests are made using HTTP/CGI for a service that is computationally intensive, the server can terminate the HTTP connection after receiving the request. Notification of results to a registered set of users is made through electronic mail. This is an example of an *asynchronous notification service*, in contrast to the synchronous service usually encountered on the Web. Such notification using electronic mail can contain URL's of result/report files. Through the use of IMAP (Internet Message Access Protocol) (IETF RFC 1730, 1996) compliant mail servers, this mail can be read and the result files retrieved using a Web browser

- **Addressing HTTP limitations: Using other protocols**

The previous discussion centered around ways of avoiding HTTP's shortcomings with respect to decision support applications. An alternative is to use protocols other than HTTP for stateful interactions. For example, applets executing in a Web browser can communicate with Java applications on the server that either provide access to or which implement the DSS using the remote method invocation (RMI) system (Arnold and Gosling, 1996) as illustrated in Figure 5.

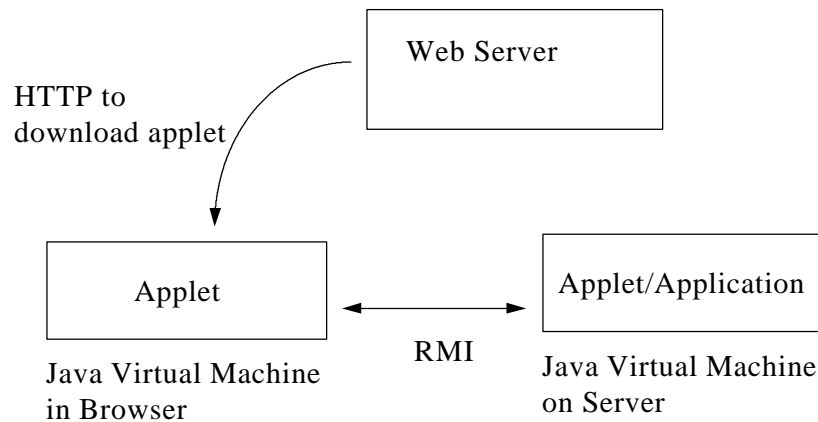


Figure 5: The Java Remote Method System

RMI permits an object executing in any Java Virtual Machine (JVM)¹ to remotely invoke methods of any other registered Java object executing in another JVM executing on another platform. Similar functionality is possible using the Internet Interorb protocol (IIOP) which is based on the common object request broker architecture (CORBA) (Siegel, 1996).

4.3 Load on Servers

In the original Web model, the Web client was limited to making HTTP requests and displaying the HTML that was downloaded from the server. When computational services are accessed, user data is collected, either online or from the client's desktop using HTML forms, and transported to the Web server for processing. Even simple error checking on forms required processing on the server. Each such request for processing using CGI invokes a new process on the Web server. This imposes a considerable load on the server and on the network, given the need for such processing in decision support applications.

¹ The Java Virtual Machine is the software implementation of a "CPU" designed to execute the byte codes that result when Java programs are compiled (Downing and Meyer, 1996).

- **Client side computing using scripting languages**

Among the first extensions to the basic Web model was the addition of computational processing capability on the Web browser. Programs written in interpreted scripting languages such as Javascript could be embedded into an HTML file as shown below.

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
document.write("An Introduction to network-centric DSS on the WWW")
</SCRIPT>
</HEAD>
<BODY>
More to come!!
</BODY>
</HTML>
```

As with applets which also enable computation on the client, the HTML file serves as a container for scripts. Programs written in Javascript can be used to considerably extend the capabilities for error checking on the client or as a stand alone programming language. In our application, Javascript could be used effectively to add to the processing of the HTML forms using which the user will supply penalty weights P_{jz} and Q for the model.

- **Client side computing using CORBA**

We discussed the use of Java applets to realize powerful interfaces in Section 4.1. Java applets are not restricted to building user interfaces. They can be used to implement the DSS engine (i.e., the model solver). Using RMI, applets can interact with other Java objects executing in remote Java Virtual Machines. However, due to security considerations, Java applets cannot access data on the client desktop. This limitation can be addressed using CORBA while retaining the platform independent features of Java. If the Java applet, the remote data source and local data objects on the client desktop are all CORBA compliant, they can communicate using the Internet Interorb protocol (IIOP). This architecture is illustrated in Figure 6.

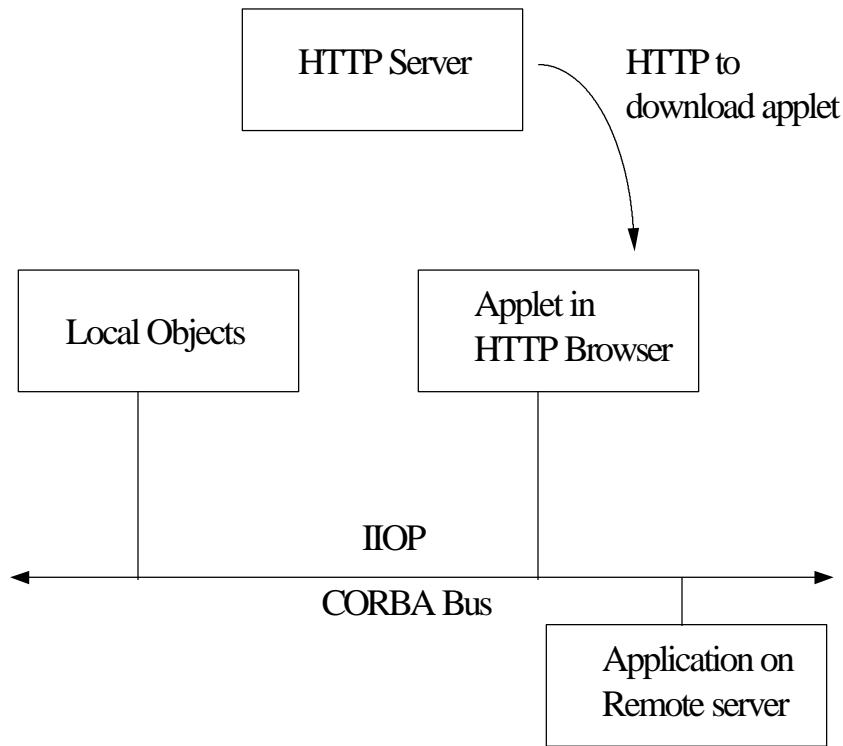


Figure 6: CORBA-based Computing

- **Client side computing using ActiveX**

Java and Javascript represent platform independent ways of making client side computing possible. ActiveX (Chappell, 1996) controls are Windows-specific software components based on the Component Object Model (COM) from Microsoft. They are binaries which are transported in HTML containers (just as Java applets are) using HTTP. Once they have been downloaded and installed, they enable all computation to take place on the client. Since several tools (e.g., databases, spreadsheet etc.) on the user desktop are Microsoft products and are ActiveX controls, a DSS application packaged as ActiveX control can interact with other user tools on the client platform.

4.5 Notification Services

HTTP requests originate at the client and are sent to the server. The server cannot communicate with the client using HTTP. This “pull” concept makes sense and gives the user a considerable amount of control. However, as discussed in the context of support requirements planning, there is a need to support three types of notification -- of results, interapplication messaging using EDI, and message initiated triggering of incremental planning. None of these notification services are supported by HTTP.

- Communication of results using “push” technologies

Over the last year, the use of the Internet to deliver “push” technology has become prevalent. One option (epitomized by Netscape’s Inbox direct service) allows an interest profile to be registered at the Web server. The server filters content from a fixed set of information services based on this profile and delivers content that is of interest to the user via email. This technology can be used to disseminate results of DSS analysis at differing levels of detail based on registered user interests.

- Inter application messaging using EDI/Email

One form of messaging that is desired is between the decision support application and other organizational applications such as force deployment (a commercial analog would be an order placement system) using EDI. Treating EDI as MIME (multipurpose internet mail extension) format, EDI messages can be sent using email as long as message formatters and processors are integrated into both the DSS application and the application with which it needs to communicate.

- Inter-application messaging using IIOP/RMI

When both the applications that need to communicate are either CORBA (Siegel, 1996) compliant or are Java objects that can be invoked remotely, IIOP or RMI are sound options. IIOP and RMI can accommodate both the EDI sort of messaging as well as the messaging required to trigger reevaluation of computation. In fact, IIOP/RMI-based messaging involves method invocation and can be easily scripted in such a manner that upon receipt of a message that data from a remote data feed has been transferred, the support requirement plans can be recomputed. This is in principle possible with EDI/Email based messaging but is considerably less efficient and

more cumbersome since it requires interacting with the email processing components (Hendry, 1993).

5. Metrics for Evaluating Architectures

Our review of Web technologies has discussed a variety of technologies which are capable of being combined to create alternative Web-based DSS architectures. We need a set of metrics to evaluate these alternatives. We propose a set of 7 metrics which are discussed below.

- Size (in bytes) of the data that needs to be transported from client to server
- Size (in bytes) of the computational application that will have to be transported from server to client to process the data at the client

When the data required by the DSS application is on the client platform, the first two metrics are used to decide if it is cheaper to transport the data to the computational application or vice versa. Of course, this also depends on whether this is a one time exercise or not.

- Quality of service

We use this term to capture the quality of the interaction (from an efficiency point of view) that the user is likely to have with a DSS application. For example, Applets and scripts in languages such as Javascript are interpreted. In contrast, ActiveX software components are binaries. Even though just in time compilers for Java have been introduced (Downing and Meyer, 1996), execution of applets is less efficient than execution of compiled ActiveX components.

- Security of the client platform

Applets and Javascript programs are not given access to the local platform. Java applets execute within the Java Virtual Machine on a Web browser. They are secure. In contrast, ActiveX software components have to be installed on the client platform and need to be trusted.

- Platform independence of the DSS application

This is a critical issue which needs to take into account the type of client platforms (e.g., Macintosh, Windows, Unix, Network devices) that will be used to access the DSS application.

Java applets and Javascript programs are platform independent. In contrast, ActiveX software components can only be used on Wintel platforms. Even if ActiveX is ported to other platforms, since they are binaries, they have to be created for each possible client platform that an end user of the DSS application is likely to possess.

- Ability to reuse existing legacy DSS implementation

A legacy application written in a language such as C/C++ cannot be easily reimplemented in a language such as Java for use on the client. However, it can be easily made available using Java on the server or using CORBA or using a gateway such as CGI. If the legacy application was a windows application it is possible to reengineer the application as an ActiveX control.

- Ability to make use of data on user desktop at the client

Both ActiveX software components or CORBA-based approaches provide this capability as described in Section 4.3.

- Support for notification

Notification is supported either using email or through the use of messaging protocols such as IIOP or RMI. In each case, basic HTTP is insufficient.

6. Implementing the Support Requirements Planning DSS

We will present two stylized DSS architectures – one server-centric and the other which emphasizes client side processing -- for the support requirements problem. We will then analyze the two alternative using the metrics introduced above.

6.1 Server-centric Architecture

The key features of this architecture are illustrated in Figure 7.

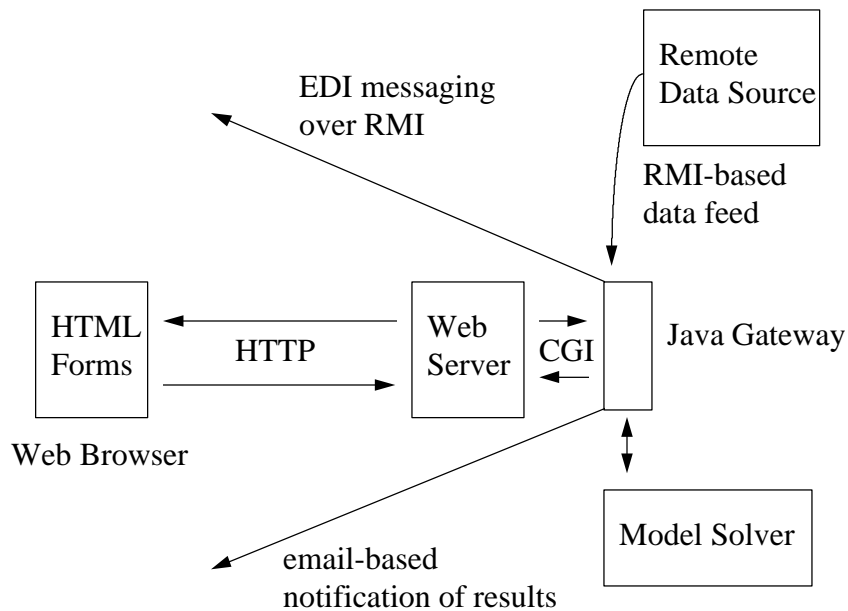


Figure 7: Server-centric Architecture for Support Requirements Planning

- User Interface

The user interface in this architecture is implemented using HTML forms. The forms-based interface enables the user to

- request execution of the model explicitly
- provide values for the penalty weights P_{jz} and Q
- execute canned queries on the database maintained on the server which is populated with data retrieved from the remote data source
- enable iterative interaction (e.g., what if analysis) with changes made to the model parameters during exploration of the decision space

Forms are processed by a Java gateway application which is invoked using CGI. Over and above these interface requirements related to supplying the data, if the user wants a direct manipulation interface, this can be implemented in this architecture by delivering an applet using HTTP to the client.

- Data Processing

The data on capabilities, C_{ijz1z2} , and demand, D_{jz} , required to instantiate the model is retrieved from the remote data source using the Java RMI system. As shown in Figure 7, the gateway is implemented in Java at the Web server. In order to use RMI for the remote data feed, the database application at the remote site has to implement a Java gateway as well. The data retrieved from the remote source is stored at the Web server as a database since this permits the user to run canned queries to extract the data set that will be used to instantiate the model. This database is also used to store static data about the combat support units and service types (the CS, CSS, and Demands set) used in the model in Figure 2.

- Algorithmic Processing

The model solver is implemented on the Web server. It is invoked using RMI by the Java Gateway. Since the model solver may be implemented in a language other than Java, a Java wrapper is needed to make this RMI-based invocation possible. Invocation of the model solver is controlled by the gateway either on receipt of a request for model solution from the user through the HTML form-based interface or based on notification from remote data source using RMI. The coordination between the data from the client, data on the server, and data from the remote data site is managed by a Java gateway.

- Notification from Remote Data Source to Java Gateway

This notification uses the Java RMI system which permits messages between any two applications executing on two Java Virtual Machines. An alternative is to use the CORBA IIOP protocol.

- Notification to Users

The results/reports from the DSS are made available to registered users using email. The email contains HTML formatted messages. These email messages can be read using Web browsers as long as users have access to IMAP compliant mail servers.

- Inter-application messaging using EDI

Inter-application messaging between the DSS and the force deployment system is also implemented using RMI to transport the EDI messages. As with the use of RMI to retrieve data

from the remote source, this assumes that the force deployment system has a Java gateway. Alternatively, one could implement EDI messaging over the CORBA IIOP protocol.

6.1.1 Analysis of Server-centric Architecture

The analysis of this architecture using the set of relevant metrics is given below.

- Quality of service to the user

The quality of service is dependent on the load on the network and the number of simultaneous sessions that are being maintained by the server. In particular it is sensitive to the connection between the remote data server and the Web server on which the DSS application is implemented.

- Security of the client platform

Excellent. This architecture can use a thin client since the data is processed at the server.

- Platform independence of the DSS application

Excellent. An end user needs to be equipped only with a Web browser. However, it assumes availability of Java on the remote data server, the Web server and on EDI messaging component. An alternative which accommodates heterogeneity of implementation languages is CORBA.

- Ability to make use of a legacy model solver implementation

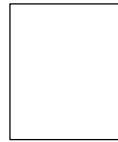
Excellent. Existing implementation can be easily integrated on to the Web server using a Java-based wrapper for the implementation.

- Support for notification

Excellent. Electronic mail integration is available on the Web server. EDI messaging components can be readily integrated on to the server as well.

6.2 Client-centric Architecture

The key features of this architecture are illustrated in Figure 8.



- Algorithmic Processing

The model solver is implemented as an Applet. Coordination with the applet used to retrieve the data is made possible using Javascript.

- Notification of Results

The results of the computation are made available to registered users using electronic mail using the features of the Web browser which serves as a mail client.

- Inter-application Messaging

EDI-based messaging uses email. While EDI formatting is implemented as an applet, email is used to transport the EDI messages.

- Coordination with the Remote Data Source

The RMI system is used to retrieve the data from the remote data server. However, this requires use of the Web server as an intermediary. Java applets can only communicate with the Web server from which they were served. Communication with the remote data source have to be routed through the Web server.

6.2.1 Analysis of the Client-centric Architecture

- Quality of service to the user

Limited by the interpreted nature of Java applet execution. Depending on the size of the applets that implement the model solver, the remote data retriever and the EDI formatter, there could be delays in downloading them using HTTP over the network.

- Security of the client platform

Excellent, since it is based on the quality of the Java security model.

- Platform independence of the DSS application

Excellent. The application uses open technologies such as Java and Javascript.

- Ability to make use of legacy DSS application implementation

Poor. For example, the model solver needs to be reimplemented in Java.

- Ability to make use of data on user desktop at the client

Not relevant here. However, if this were required, it can be implemented using the CORBA computing architecture illustrated in Figure 6.

7. Conclusions

We have presented a review of Web technologies of relevance to DSS researchers and developers. Using the information processing requirements of Support Requirements Planning, we have presented and analyzed a server-centric and client-centric architecture. While we used Java-based technologies to illustrate implementation of these architectures, there are other alternatives. For example, ActiveX based technologies could be used to realize the client-centric architecture. While this is currently a Windows platform-specific technology, it is superior to the Java-based proposal on the quality of service metric and if integration with data on the client platform is an important requirement.

Topics at the intersection of information networking and DSS have recently generated a lot of interest in the DSS community (Bhargava, Krishnan, Muller, 1997a, 1997b, Ba et. al., 1995; Juesfeld and Bui, 1995). With these researchers, we believe that Web-based technologies represent a major opportunity to expand the reach of DSS technology, both to distributed groups of end users as well as to new classes of applications characterized by the need to incorporate streaming, remote data feeds and integration with other organizational applications. We hope that this paper will further increase interest in this important topic.

Acknowledgements: This paper was funded in part by the National Science Foundation (grant IRI-9312143) and the U.S. Army Artificial Intelligence Center (grants DAAH04-94-G-0239 and DAAH04-96-1-0385) through the Army Research Office. We would also like to thank Maj. William Branley (Retd.) and other members of the AI Center for several discussions related to the Support Requirements Planning problem. The opinions expressed in the paper reflect our understanding of the problem and are not to be attributed to the funding agencies.

References

1. Arnold, K., Gosling, J., **The Java Programming Language**, Addison Wesley Publishing Company, 1996.
2. Ba, S., Kalakota, R., Whinston, A., Executable documents as the basis for DSS, Proceedings of the Third ISDSS Conference, Hong Kong, 373-381, 1995.
3. Bhargava, H.K., Krishnan, R., The World Wide Web: Opportunities for OR/MS, Feature Article in Preparation, *INFORMS Journal on Computing*.
4. Bhargava, H., Krishnan, R., Muller, R., Decision Support on Demand: Emerging Electronic Markets for Decision Technologies, *Decision Support Systems*, Forthcoming, 1997a.
5. Bhargava, H., Krishnan, R., Muller, R., Electronic Commerce in Decision Technologies: A Business Cycle Analysis, *International Journal of Electronic Commerce*, Forthcoming, 1997b.
6. Bradley, S. Hax, A., Magnanti, T., **Applied Mathematical Programming**, Addison Wesley Publishing, 1977.
7. Berners-Lee, T., Cailliau, R., Luotinen, A., Nielsen, H., Secret, A., The World Wide Web, *Communications of the ACM*, 37:8, 76-82, 1994.
8. Chappell, D., **Understanding ActiveX and OLE**, Microsoft Press, 1996.
9. Downey, T., Meyer, J., **The Java Virtual Machine**, McGraw Hill, 1996.
10. ElMasri, H., Navathe, S., **Fundamentals of Database Systems**, Benjamin Cummins Publishing Company, 1994.
11. Garey, M., Johnson, D., **Computers and Intractability: A Guide to the Theory of NP-Completeness**, W. H. Freeman, 1979.
12. Glover, F., Tabu Search, Part I, *ORSA Journal on Computing*, 1:3, 190-206, 1989.
13. Glover, F., Tabu Search, Part II, *ORSA Journal on Computing*, 2:1, 190-206, 1990.
14. Goldberg, D. E., **Genetic Algorithms in Search, Optimization, and Machine Learning**, Addison Wesley, Reading, MA, 1989.
15. Hendry, M., **Implementing EDI**, Artech House Publisher, 1993.
16. Holland, J. **Adaptation in Natural and Artificial Systems**, University of Michigan Press, Ann Arbor, 1975.

17. IETF RFC 1730, The Internet Message Access Protocol,
<http://andrew2.andrew.cmu.edu/rfc/rfc1730>
18. Jeusfeld, M., Bui, T., Interoperable Decision Support System Components on the Internet, Proceedings of the WITS Conference, 56-67, 1995.
19. Jones, C. V., Developments in Graph-Based Modeling for Decision Support, *Decision Support Systems*, 13:1, 61-74, 1995.
20. Kimbrough S., Pritchett, C.W., Bieber, M.P., Bhargava, H.K., The Coast Guard's KSS Project, *Interfaces*, 20:6, 5-16, 1990.
21. Krishnan, R., Support Requirement Planning: The ATLAAS Project Report, Report to the US Army Artificial Intelligence Center, 1995.
22. Michalewicz, Z., **Genetic Algorithms + Data Structures = Evolution Programs**, Springer-Verlag, 1992.
23. Orfali, R., Harkey, B., Edwards, J., **Essential Client/Server Survival Guide**, John Wiley and Sons, Inc., 1994.
24. Parker, M., **Strategic Transformation and Information Technology**, Prentice Hall, NJ, 1996.
25. Reeves, C., **Modern Heuristic Techniques for Combinatorial Optimization**, Ed. C. Reeves, Halstead Press, 1993.
26. Reimer, D., FORSCOM's Use of Technology in the 21st Century, *Army*, 45-50, May 1994.
27. Siegel, J., **CORBA Fundamentals and Programming**, John Wiley and Sons, 1996.
28. Van Herwijnen, E., **Practical SGML**, Kluwer Publishers, 1994.
29. Yeager, N., Mcgrath, R., **Web Server Technology**, Morgan Kauffman Publishers, 1996.