

Air Force Institute of Technology

**AFIT Scholar**

---

Theses and Dissertations

Student Graduate Works

---

3-2000

## A Methodology for Integrating Tools in a Web-based Environment

Musa Serdar Arslan

Follow this and additional works at: <https://scholar.afit.edu/etd>



Part of the [Computer Engineering Commons](#)

---

### Recommended Citation

Arslan, Musa Serdar, "A Methodology for Integrating Tools in a Web-based Environment" (2000). *Theses and Dissertations*. 4736.

<https://scholar.afit.edu/etd/4736>

This Thesis is brought to you for free and open access by the Student Graduate Works at AFIT Scholar. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of AFIT Scholar. For more information, please contact [AFIT.ENWL.Repository@us.af.mil](mailto:AFIT.ENWL.Repository@us.af.mil).



**A METHODOLOGY FOR INTEGRATING  
TOOLS  
IN A WEB-BASED ENVIRONMENT**

**THESIS**

Musa Serdar ARSLAN, 1<sup>st</sup> Lt. TUAF

AFIT/GCE/ENG/00J-01

**DEPARTMENT OF THE AIR FORCE  
AIR UNIVERSITY  
AIR FORCE INSTITUTE OF TECHNOLOGY**

**Wright-Patterson Air Force Base, Ohio**

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

20000815 185

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense, the U. S. Government, or the Government of the Turkish Republic.

---

A METHODOLOGY FOR INTEGRATING TOOLS  
IN A WEB-BASED ENVIRONMENT

THESIS

Presented to the Faculty of the Graduate School of Engineering and Management  
of the Air Force Institute of Technology  
Air University  
in Partial Fulfillment of the  
Requirements for the Degree of Master of Science in Computer Engineering

Musa Serdar ARSLAN, B.S. Computer Engineering

1<sup>st</sup> Lt. TUAF

April 2000

A METHODOLOGY FOR INTEGRATING TOOLS  
IN A WEB-BASED ENVIRONMENT

Musa Serdar ARSLAN, B.S. Computer Engineering

1<sup>st</sup> Lt. TUAF

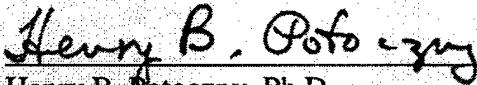
Approved:



Thomas C. Hartrum, Ph.D.  
(Chairman)

April 18, 2000

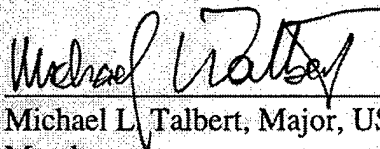
Date



Henry B. Potoczny, Ph.D.  
Member

April 18, 2000

Date



Michael L. Talbert, Major, USAF, Ph.D.  
Member

20 Apr 2000

Date

## Acknowledgements

AFIT education was a very challenging experience for me. We have always had to work very hard. I won't be able to thank my family enough for always supporting me. I wouldn't be able to overcome this challenging education period without the support of my family and my friends. My wife Funda has always been supporting me, and my little daughter Melike has always been there to take me away from homework and projects for a little time. I also want to thank my friends who always have been supporting and helping me. I would like to thank my thesis advisor, Dr. Hartrum, for always encouraging and guiding me. He helped me to improve my writing skills in English and he always pushed me forward to make it better. I also want to thank my thesis committee members Dr. Potoczny and Maj. Talbert for always helping me when I needed their help and for their valuable comments on my thesis. I also want to thank my classmates who have been good friends and who showed good examples of American culture. I also want to thank the Turkish Air Forces who gave me this opportunity and all the Turkish citizens who support me by paying their taxes. Thank you all.

Musa Serdar ARSLAN

## TABLE OF CONTENTS

<i>Acknowledgements</i> .....	<i>i</i>
<i>Abstract</i> .....	<i>viii</i>
<b>1. Introduction</b> .....	<b>1</b>
<b>1.1. Background</b> .....	<b>1</b>
<b>1.2. Problem</b> .....	<b>2</b>
<b>1.3. Summary of Prior Effort</b> .....	<b>5</b>
<b>1.4. Scope</b> .....	<b>6</b>
<b>1.5. Approach</b> .....	<b>6</b>
<b>1.6. Assumptions</b> .....	<b>7</b>
<b>1.7. Thesis Overview</b> .....	<b>7</b>
<b>2. Web-Based Technologies</b> .....	<b>9</b>
<b>2.1. Server-Side Technologies</b> .....	<b>11</b>
2.1.1. CGI (Common Gateway Interface).....	11
2.1.2. Java Servlets.....	12
2.1.3. Active Server Pages (ASP) .....	13
2.1.4. Java Server Pages (JSP) .....	14
<b>2.2. Client-Side Technologies</b> .....	<b>14</b>
2.2.1. Java Applets .....	15
2.2.2. ActiveX Controls.....	17

2.2.3. Dynamic HTML (DHTML) .....	18
<b>2.3. Database Connectivity .....</b>	<b>18</b>
2.3.1. JDBC .....	18
2.3.2. ODBC .....	22
<b>2.4. Previous Work .....</b>	<b>24</b>
<b>2.5. Summary .....</b>	<b>28</b>
<b>3. <i>Web-based Integration Methodology</i>.....</b>	<b>29</b>
<b>3.1. Overview .....</b>	<b>29</b>
<b>3.2. Analysis Of The Tools .....</b>	<b>30</b>
<b>3.3. Integration Methods.....</b>	<b>35</b>
<b>3.4. Choosing An Integration Method .....</b>	<b>38</b>
<b>3.5. Analysis Of The Target Platform .....</b>	<b>42</b>
<b>3.6. Design of the System .....</b>	<b>47</b>
<b>3.7. Summary .....</b>	<b>53</b>
<b>4. <i>Application of Methodology</i>.....</b>	<b>54</b>
<b>4.1. Analysis Of The Tools .....</b>	<b>54</b>
<b>4.2. Choice Of Integration Method .....</b>	<b>59</b>
<b>4.3. Analysis Of The Target Platform .....</b>	<b>60</b>
<b>4.4. System Design .....</b>	<b>62</b>
<b>4.5. Summary .....</b>	<b>70</b>



<b>5. Results and Conclusion.....</b>	<b>71</b>
<b>5.1. Results .....</b>	<b>71</b>
<b>5.2. Conclusions .....</b>	<b>73</b>
<b>5.3. Recommendations For Future Work .....</b>	<b>73</b>
5.3.1. Recommendations For The Methodology.....	74
5.3.2. Recommendations For The Course Registration System.....	75
<b>Appendix A. Table Definitions.....</b>	<b>76</b>
<b>Bibliography .....</b>	<b>78</b>
<b>Vita.....</b>	<b>81</b>

## LIST OF FIGURES

<i>Figure 1.1: Current course registration process.</i> .....	4
<i>Figure 2.1: Web-based applications.</i> .....	10
<i>Figure 2.2: The ODBC Architecture</i> .....	23
<i>Figure 3.1: Interaction of the user with the tools before integration.</i> .....	35
<i>Figure 3.2: The first integration paradigm.</i> .....	36
<i>Figure 3.3: The second paradigm</i> .....	37
<i>Figure 3.4: The third paradigm</i> .....	38
<i>Figure 3.5: A Generic Internet Platform</i> .....	43
<i>Figure 3.6: System design using the first paradigm and Tool A as front end.</i> .....	48
<i>Figure 3.7: System design using the first paradigm and Tool A is distributed between the client and the server.</i> .....	50
<i>Figure 3.8: System design using the first paradigm with a newly designed front and back end.</i> .....	51
<i>Figure 3.9: System design using the second paradigm.</i> .....	52
<i>Figure 3.10: System design using the third paradigm.</i> .....	53
<i>Figure 4.1: First level DFD of database administration applet.</i> .....	55
<i>Figure 4.2: First level DFD of course registration applet.</i> .....	56
<i>Figure 4.3: First level DFD of Edplan Checker tool.</i> .....	57
<i>Figure 4.4: System design using the first paradigm.</i> .....	60
<i>Figure 4.5: Design of the integrated system using the first paradigm and the first tool on the client side.</i> .....	63
<i>Figure 4.6: Main database administration page</i> .....	64

*Figure 4.7: Course registration applet window*..... 65

## LIST OF TABLES

<i>Table 3.1: Steps of the integration methodology.....</i>	<i>30</i>
<i>Table 3.2: Choosing the integration paradigm.....</i>	<i>38</i>
<i>Table 3.3: Choice of integration paradigm according to the extendibility method.....</i>	<i>40</i>
<i>Table 4.1: Tool Characteristics Summary .....</i>	<i>59</i>

## **Abstract**

Web-based technologies are evolving very rapidly. New technologies are introduced very frequently in the realm of the Web and Internet. This evolution also affects database management systems (DBMSs). Almost all DBMS vendors are making their systems "Web-enabled." The Internet and the World Wide Web are getting more important and bigger than ever. Because of the increase in the importance of the Internet and the Web, migrating old applications and tools to a web-based environment is becoming more important. When migrating old applications or tools to the web-based environment, integration of tools becomes an important issue. In this research, a step-by-step methodology for integrating tools in a web-based environment is created while trying to integrate two tools that are part of the AFIT education plan administration and course registration system. These two tools are CLASPICS (Computerized, Lightweight Assistant for Student Programme Identification and Course Selection) and Education Plan Checker tool. The methodology proved to be favorable for integrating tools in a web-based environment. In the results and conclusion part of the thesis some recommendations for improving the methodology and the Edplan administration and course registration system are given.

# A METHODOLOGY FOR INTEGRATING TOOLS IN A WEB-BASED ENVIRONMENT

## 1. Introduction

### *1.1. Background*

The software technology in the realm of database systems and Internet computing is evolving very rapidly. We hear of new technologies announced every week on both the hardware and the software side. The evolution of the database systems is making its way towards the Internet. DBMS vendors are trying to make their products "Web-Enabled". Some DBMSs are now usable and maintainable from within a browser, which can be run on any platform. They are trying to add new functionality to their software to supply easy web publishing of the data in the database. They are trying to add capabilities like dynamic web publishing directly from the database. Because all business and work are moving to the Internet, companies and organizations are also trying to migrate their old legacy databases and applications to the Web environment, or they are trying to make them web enabled. Instead of discarding the old applications and databases, companies or organizations "web-enable" their applications and migrate their systems into a web-based system. Rebuilding the system from scratch may be another option, but it should be the last option because of its cost. Creating a new system and software will obviously be more costly than migrating old applications in a web-based environment. While migrating or web-enabling the applications, integration of the tools becomes an important issue. All the companies or organizations would like to integrate the tools they use and take full advantage of the web-based environment. While web-enabling applications and tools, we may eventually need to integrate tools.

The Air Force Institute of Technology (AFIT) also has a big web-based environment. This is a big intranet. There are lots of different tools and applications running on this network. These are either commercial applications or tools created by students in class projects. Each of these tools does many different tasks. Some of them are used for similar tasks and they do redundant work. They store the same information in different places and different formats. They use different database systems. Some of them use the same database system, but still store the same information repetitively. Integrating the related applications and tools will solve some problems and get rid of storing redundant information. This will centralize the information and make it more consistent.

In such a web-oriented world, we are using a course enrollment system at AFIT that is time consuming and requires redundant work. We have many facilities in AFIT to make use of for creating a better and more efficient system. We can make full use of the web-based environment in AFIT by creating a more coherent system. Former work in this area resulted in some good solutions for some of the problems. But all the work done so far takes just one part of the problem, and tries to give a solution for that part. Each task has a different approach to the problem. We have to make a web-based and complete system, which includes all other work done so far. Having this web-based environment, we can integrate all the tools and solutions created until now on this environment to make a more consistent and coherent system as a whole.

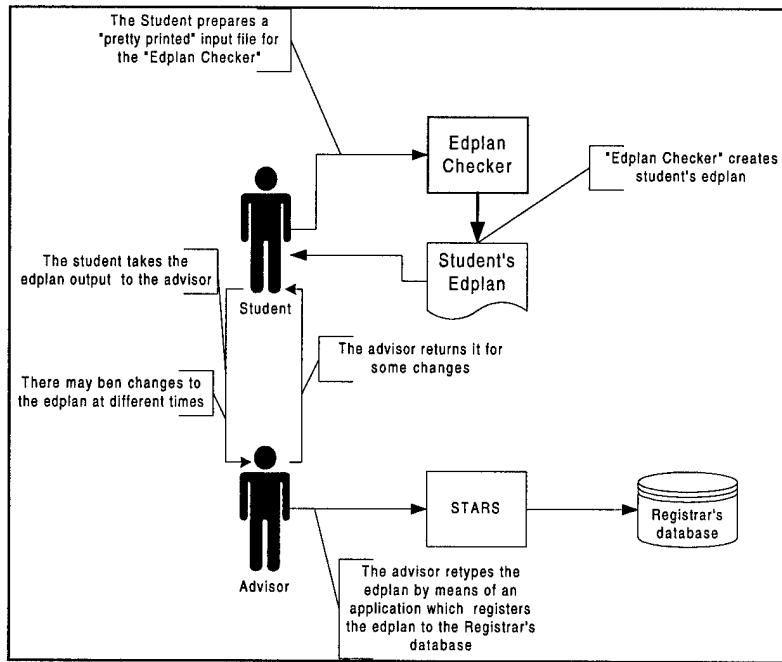
## ***1.2. Problem***

The current enrollment system works as follows. The student decides which courses to take according to his/her programme requirements, choices, and the recommendations of his/her advisor. After preparing the education plan (Edplan) manually,

he/she prepares a formatted input file for an Edplan Checker tool using a text editor. This tool reads the input file and checks the student's Edplan for any errors and for compliance with the AFIT requirements. Many problems occur in this step. Generally, students make many typing errors when preparing the input file for the Edplan Checker. Because of these typing errors, running the Edplan Checker tool becomes hard and time consuming. The student edits the input file and corrects the typing error. If there aren't any other typing errors, the Edplan Checker tool finally checks the student's Edplan. Then it gives an output file that includes which courses the student takes each quarter, the credit hours for each course, total credit hours for each quarter, overall total credit hours, and a classification of courses. Then the student gives a printout of this file to the advisor. The advisor checks the draft Edplan for any possible changes. If the advisor rejects the Edplan then the student resubmits it after making necessary changes. After the final Edplan is submitted the advisor inputs the Edplan of the student to the actual course registration database via a client-side application (STARS) by retyping all the courses. This phase is the time consuming part for the advisor. The advisor has to input courses for each quarter for each of his students. If any change is needed for the Edplan of the student, the same things are done again. This process is shown in Figure 1.1.

There may be other changes to the Edplan of the student later, any time during the education period. The student may want to make changes to his/her Edplan before a quarter starts. The student can also drop or add some courses after the beginning of a quarter. There may be changes to the Edplan any time during the education. The student may need to change his/her Edplan due to some course cancellations. Because of these





**Figure 1.1: Current course registration process.**

frequent changes to the Edplan the process of making the Edplan, entering it into the Registrar's official database, and changing it has to be easy and done without redundancy.

As a recent improvement to the system, a tool for Edplan creation and maintenance (CLASPICS – Computerized Lightweight Assistant for Student Programme Identification and Course Selection) is created by Jason Gunsch, a summer intern student. This tool helps the students while preparing their Edplans and lets them to save the Edplans in a database for future use and reference. This tool also creates and displays the input file for the Edplan Checker tool. But the student has to cut and paste this output of the tool to a text file and save it. After that the student has to run the Edplan checker tool with this input file. This solves the creation of the input file for the Edplan checker tool to some extent. But it is desirable that the student could be able to prepare the Edplan and check it from one tool.

## PROBLEM STATEMENT:

The actual problem is to define a methodology for integrating existing tools into a web-based system. By doing so, we will be able to integrate the existing tools into a web-based system and to develop a prototype enrollment system at AFIT to demonstrate the methodology, which provides ease of use to the users, prevents typing errors, and removes the redundancy in the current system.

### ***1.3. Summary of Prior Effort***

Tien-Chen Lee's [1] thesis project was to develop a web-based prototype for AFIT Edplan administration. In his project, he follows the "Engineering Software Components for Web-Based Access" methodology of Daniel DiPiro [2] to develop a prototype system. After providing some background information on some database connectivity techniques (such as CGI, ODBC, JDBC, and ADO), and software technologies for web-based database access (Applets, ASP), he applies DiPiro's methodology for the prototype system. He uses Microsoft's ASP, and ADO technologies to implement his system.

Another work on the same problem was done by Penelope Noe as a special study. She developed an application that inputs the Edplan directly to Registrar's database using the input file that the student created for the Edplan checker tool. She uses SQL and C++ to directly input the data into the Registrar's database.

The work done by Jason Gunsch makes big contributions to solve the problem. He created a tool for the students to create and save their education plans. This tool lets the students choose the courses from a list of courses using a graphical user interface (GUI). This tool eases the Edplan preparation process and reduces the time spent for the

process by the student. Since the tool saves the Edplan of the students in a database, the students can easily make changes to the Edplans and save them again. The tool also lets the advisors see the Edplans of their students and make any change on them by loading from the database. The tool creates a requirements structure in the database and helps the students to easily create their Edplans without any typing errors. However, the Edplan created by this tool still has to be checked by the Edplan Checker tool. This tool helps the students at this point also by creating the input file for the Edplan Checker tool.

### ***1.4. Scope***

The main objective of this research is to develop a general methodology for integrating tools in a web-based environment. This project does not include creating a course scheduling system and integrating it to the current system. The main concern will be to demonstrate the methodology by creating an easy to use web-based Edplan creation and administration system to be used by the students and the advisors by integrating the existing tools.

### ***1.5. Approach***

While doing this project the following approach was used:

1. Learn ways of developing web-based applications:

Survey the current software engineering approaches for web-based and distributed application development to decide on an appropriate analysis/modeling system.

2. Analyze the current system:

Figure out how the current system works and find out all the problems with the current system and the requirements for the new system.

3. Analysis and modeling:

Perform the analysis and modeling techniques used for development of a web-based application.

4. Implementing the created model:

Implement the created model in the selected platform.

5. Testing the developed application:

Test the developed application and show its usability.

6. Define a general methodology based on the enrollment system project

### ***1.6. Assumptions***

Before beginning the methodology and system development process, a few assumptions have to be made. It is assumed that there are enough resources for gaining enough knowledge about the tools and development environments that will be used while developing the methodology and developing the system. It is also assumed that the SC department will provide any necessary schema definition and interface documentation for the actual Registrar's database. Another assumption is that all necessary resources like hardware and software resources, past research, and documentation are available.

### ***1.7. Thesis Overview***

This chapter gives the description of the problem to be solved, lists the previous work done about the problem and related subjects, gives the scope of the thesis investigation, and the approach taken in solving the problem. Chapter II is a literature review of basic web and Internet technologies, and presents an overview of the research done previously about the subject. In Chapter III, the "web-based integration methodology" is ex-

plained. In Chapter IV, an application of the methodology is demonstrated on two tools to solve the problem. The last chapter presents the results and conclusions of this effort.

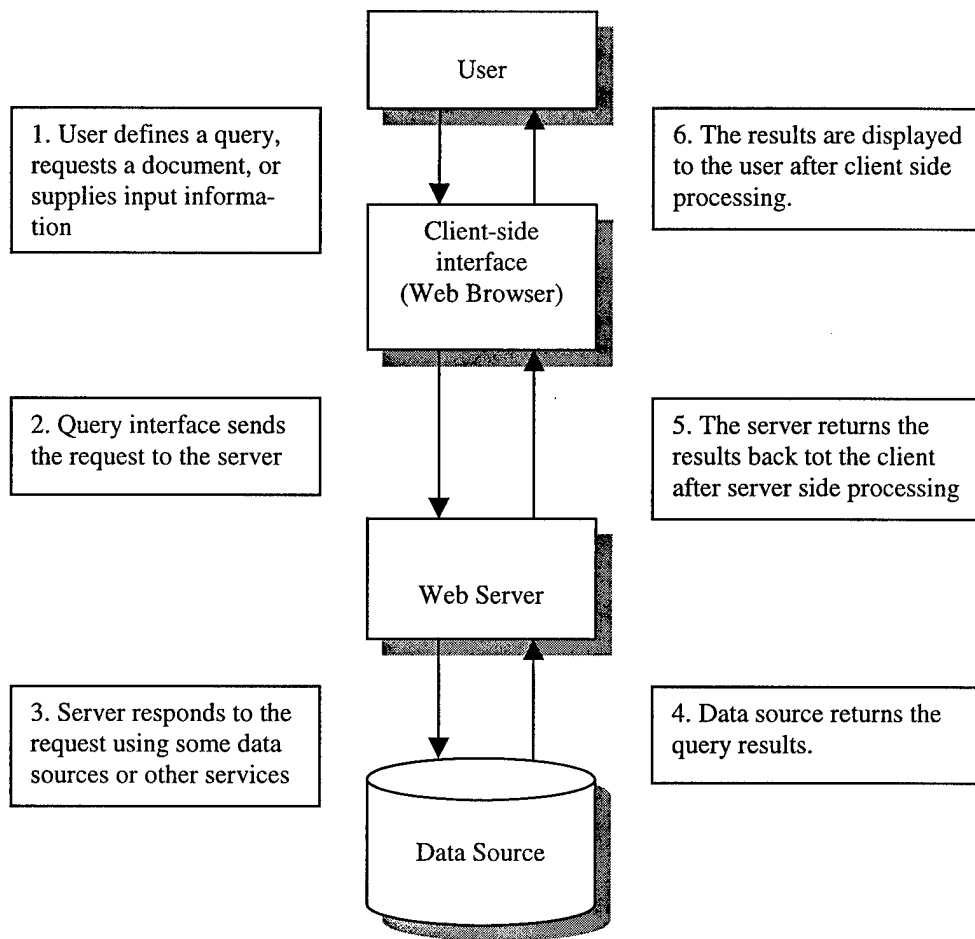
## 2. Web-Based Technologies

There are many different web-based application development technologies. However, they can be used interchangeably. While developing web-based applications, there is no right way to solve a problem and there is no truly unique technology. However, many tools and technologies can be configured in multiple ways to solve the same set of problems. For example, server side technologies, like Common Gateway Interface (CGI), Active Server Pages (ASP), Java servlets, Java Server Pages (JSP), and other proprietary server-side technologies all serve the same basic function. Similarly, Common Object Request Broker Architecture (CORBA), Distributed Component Object Model (DCOM), Enterprise Java Beans (EJB), and Remote Method Invocation (RMI) all specify how distributed objects communicate. Of course, they have many different properties and they have advantages and disadvantages over each other.

As such, web technologies within the same categories can be used interchangeably in most situations. I will explain some web technologies that can be used to develop web-based applications in this chapter.

Web-based applications generally do similar things. These are shown in Figure 2.1 and are listed as follows:

- Provide a user interface: Web applications have to provide users an interface for entering data or requesting data. This can be a web browser with an HTML form loaded or a Java applet running on it.
- Transmit the Data or Request: The user data or request is sent to the web server. This is generally done using HTTP protocol.



**Figure 2.1: Web-based applications.**

- **Perform Server Side Processing:** The web server processes the user data using a middleware. This can involve sending the user query to a data source and getting the results, applying some business logic to the user data, and/or server-side processing of the requested document.
- **Transmit the Results Back:** The processed data is sent back to the client if it was requested, by using HTTP protocol again.
- **Perform Client Side Processing:** Finally the returned data is processed on the client side again to be displayed correctly. This can be the interpreta-

tion of an HTML page, possibly with some embedded scripts like JavaScript or VBScript, or displaying the data in a Java applet.

## **2.1. Server-Side Technologies**

### **2.1.1. CGI (Common Gateway Interface)**

The Common Gateway Interface is a standard that specifies communication between HTTP servers and server-side gateway programs. When a gateway program is requested from the server, the server activates the program and passes the data from the client to the program. The program processes the data and sends the result back to the server. Then the server sends this data back to the client. The format of the data passed back and forth between the server and the gateway program is defined by the CGI specification [22].

CGI Scripts are easy to implement; they don't require knowledge of any new programming environment. They can be implemented in any standard programming language like C, C++, or Pascal. There is no need to change the programming language currently used to develop a CGI application. CGI applications can also be scripts written in scripting languages like perl, tcl, or other shell programs. CGI applications are also very common and supported by almost all of the web servers on the market.

Since CGI has some performance limitations, it is not suited for large-scale systems. This is because every request starts a new process on the server. In addition, CGI scripts, usually written in perl, lack many features of a more general programming language like Java. However, CGI is well suited for small-scale projects with low performance requirements.



### **2.1.2. Java Servlets**

Java servlets are compiled Java code that extend Java-enabled web servers. Java servlets are starting to be used more than the CGI scripts. Since the servlet API, which is used to write servlets, assumes nothing about the server's environment or protocol, servlets can be embedded in many different servers. Almost all of the web servers on the market today have support for Java servlets. A list of the web servers that support Java servlets can be found in Sun's Javasoft web pages [4].

Java servlets are to the server what applets are to the browser. Anybody who knows how to program in Java, can write Java servlets. All he/she needs to do is use Sun's Java Servlet API. Since the servlets are pure Java, they carry all the advantages of the Java programming language. Java servlets are inherently platform independent. You can run a servlet on any web server running on any platform without making any changes. There has to be a Java Virtual Machine (JVM) running on the server to run the servlets. This leads to some other benefits. The built-in garbage collection won't allow any resource leak caused by servlet errors or exceptions. Java servlets give a robust developing environment to the web developers. They can use the full power of a computationally complete and platform independent programming language like Java. Servlets obtain database access using the JDBC API. Distributed programming is also supported using Remote Method Invocation (RMI) or CORBA. Java servlets can be used to get rid of security issues while accessing an RDBMS, which is running on a machine other than the web server. Java servlets are becoming the most robust and widely used server side technology. [17]

### **2.1.3. Active Server Pages (ASP)**

Active Server Pages (ASP) technology is a powerful web server feature of Microsoft. ASP was introduced with Microsoft's Internet Information Server (IIS) version 3 [23]. ASP technology allows mixing HTML and server-side scripting in a file. The client requests an ASP file from the server. The web server processes the file with the *.asp* extension before sending it to the client. After processing the script embedded in the HTML code in the ASP file, the server creates a dynamic html page for the client and sends it back. By using ASP technology, developers can use Microsoft's ActiveX Data Objects (ADO) to access different data sources using ODBC. Two different scripting languages can be used in ASP files. These scripting languages are VBScript and Jscript. VBScript is a stripped-down version of Microsoft's Visual Basic programming language and Jscript is the Microsoft implementation of Netscape's JavaScript server-side scripting language [24]. The combinations of scripting language, programmatic access to ActiveX components, and dynamic generation of HTML make ASP a powerful technology for building dynamic web sites [25]. ASP can link to back-office services and technologies including Microsoft's COM (Component Object Model) objects and Microsoft Transaction Server (MTS) at the server side to build complicated business logic [26]. Using ASP is easy compared to CGI scripts. ASP is designed as a convenient alternative to conventional CGI scripting using perl or C scripts.

The biggest drawback of ASP technology is that it is Microsoft specific and a platform dependent technology. Only IIS has support for ASP and IIS runs only on Windows NT platforms. There are some recent efforts to give ASP support on the UNIX platform. For example, Chilisoft [5] has developed ChiliASP to give ASP support on a Unix platform. There is also an ASP module for the widely used web server Apache.

#### **2.1.4. Java Server Pages (JSP)**

JSP is Sun Microsystems' technology that mirrors Microsoft's ASP. Like ASP, JSP lets users embed code directly into HTML pages for server-side processing and dynamic web-content delivery. It uses a similar technology, but instead of using scripting languages, leverages the full power of the Java programming language. JSP can make use of Java servlets and Enterprise Java Beans (EJB) on the server side like ASP does with ActiveX objects. [26]

JSP handles the code embedded in HTML in a different way than ASP. ASP interprets the embedded script in the HTML. JSP compiles the Java code into a servlet and then executes it by the Java Virtual Machine. The compilation of the code occurs only once on the first request of a JSP page. [29]

### ***2.2. Client-Side Technologies***

Client side scripting languages are designed to allow programs to be encapsulated in HTML documents as plain text. These small programs are executed on the client machine by the browser. These scripts perform computations and react to events on the client processor. The main purpose of client side scripting is to use the CPU power on the client computer, which does not do anything other than browsing the web. It is a good idea to share some of the burden between the client and server by taking some of the processing load off the server and giving it to the client. [27]

Client side scripting is useful to do things like maintaining state, filling out forms, error checking, checking validity of data input in an HTML form, or doing numeric calculations. The client browser doesn't need to go back to the server to do these things. By using client side scripting, we can take some of the workload off the server and give it to

the client. We can also eliminate some network traffic. Client side applications maintain security by keeping server processing to a minimum. They are not restricted by HTTP and very good GUI designs can be done. [27]

The most used client-side scripting languages are JavaScript (Netscape), JScript, and VBScript (Microsoft) [27]. JavaScript was first developed by Netscape and used in their browser Navigator 2.0. JScript and VBScript are Microsoft's scripting languages used in Internet Explorer.

The biggest problem with the client side scripting is the lack of standards. Because not all the browsers have support for all scripting languages, web pages that include scripts generally cannot be loaded and run correctly on all browsers. To get rid of this problem ECMA (European Computer Manufacturers Agency), a standardization organization developed a standard for JavaScript. This standard was named as ECMAScript and documented in the ECMA-262 specification. The ECMA-262 standard is also approved by ISO (International Standardization Organization) as ISO-16262. The current versions of both JavaScript and JScript have support for ECMAScript. [28]

### **2.2.1. Java Applets**

Java applets are downloaded client side programs that run within the Java Virtual Machine (JVM) of a Java-enabled web browser. The name of the Java applet is embedded in the HTML code via an <APPLET> tag. When a browser loads an HTML page with a Java applet it loads the Java applet from the server and runs it in its JVM. [6]

By using Java applets developers can use the full power of the Java programming language within the security limits of the browser. Applets are restricted to functioning within the Java security framework. If an applet is not signed by a trusted source, it can

only run in a secure sandbox within the browser JVM. If an applet is not signed, many restrictions apply to it [6]:

- Applets cannot access the local file system of the client computer.
- Applets cannot create a network connection to a machine other than the one from which the applet was loaded.
- Applets cannot act as network servers, listening for or accepting socket connections from remote systems.
- Applets are prevented from executing any programs that reside on the local computer.
- Applets are not allowed to load dynamic libraries or define native method calls.
- Applets are allowed to read only certain system properties and are prevented from accessing others. (Within the Java environment, various standard system properties are set. These properties can be accessed with the `java.lang.System.getProperty(String key)` method.)
- Applets cannot manipulate any Java threads other than those within their own thread group.
- Applets cannot shut down the JVM. (By calling `System.exit()`)
- Applets cannot create a `SecurityManager` or `ClassLoader` instance. The Java browser creates such an object and uses it to impose the security policy on all applets.
- The `java.net` package uses factories to establish particular implementations of specific concepts: protocol handlers, content handlers, and sockets.

Applets cannot override the `java.net.ContentHandler-Factory`, and `java.net.SocketImplFactory`.

### 2.2.2. ActiveX Controls

ActiveX controls are reusable programs that can be inserted into a web page or other application. ActiveX controls are native Windows applications and therefore run only within these environments. ActiveX controls communicate with the browser and the operating system using a well-defined COM interface.

Developers can insert ActiveX controls into a Web page by using the `<OBJECT>` tag in the HTML page and referring to the **ClassId** of the ActiveX control. When a user clicks on a page with an OBJECT tag and a **ClassId**, ActiveX on the client looks up the client systems registry to find the ActiveX control code referred to with the **ClassId**. If the relevant ActiveX control is already present on the client system, that code is invoked and executed on the client. If the ActiveX control code is not already present on the client system, it is downloaded from the server, installed on the client after checking for security, and then invoked on the client. Once downloaded and installed, an ActiveX control can be used in any HTML page or any Windows application. ActiveX will also provide versioning support by enabling the option to download the latest version of an ActiveX control.

ActiveX controls are Windows-specific programs. They don't work on other operating systems. They also cannot work on a browser other than Internet Explorer without a plug-in. Developers can use many different kinds of development environments and languages, like Visual C++, Visual Basic, Visual J++, Borland Delphi, Borland C++, and Symantec C++. ActiveX controls created using these languages can do everything that tradi-

tional Windows applications can do, including all native operating system services. This gives great power to the developers.

### **2.2.3. Dynamic HTML (DHTML)**

Dynamic HTML is another technology that adds action to static web pages. The basic notion of DHTML is to allow any element of a web page to be changeable at any time. With DHTML, all the work is done on the client side. All page modifications appear immediately following a trigger, such as a user selection, or a mouse click. All the aspects of a loaded page including text styles, swapped images, context-sensitive forms and tables, and the data can be modified. DHTML describes the abstract concept of breaking up a page into manipulable elements, and exposing those elements to a scripting language that can perform manipulations [7].

## **2.3. Database Connectivity**

Currently there are two types of database connectivity methods widely used on web-based applications and on the Internet platform. These are ODBC (Open Database Connectivity) and JDBC (Java Database Connectivity). ODBC is developed by Microsoft and is mostly used by Microsoft technologies. JDBC is developed by Sun Microsystems and is used by Java applets or Java applications.

### **2.3.1. JDBC**

JDBC is a Java API for executing SQL statements. It consists of a set of classes and interfaces written in the Java programming language. JDBC provides a standard API for tool/database developers and makes it possible to write database applications using a pure Java API. The JDBC API is based on the X/Open CLI (Call-Level Interface), which

defines how clients and servers interact with one another when using database systems. One of the most important benefits of using JDBC API is database interoperability. The program developer can change the underlying database driver without having to modify the application. There is a JDBC-ODBC bridge driver that comes with the standard Java programming API. This bridge driver can be used in Java programs to access ODBC databases. [30]

There are four basic types of JDBC drivers. These four types can be listed as follows:

*Type 1: The JDBC-ODBC Bridge.* The JDBC-ODBC bridge is provided by JavaSoft as part of its JDK (Java Development Kit). This bridge is part of the sun.jdbc.odbc package and is not required to be ported by vendors that provide a Java virtual machine. This bridge uses native ODBC methods and has some limitations in its use. This driver can be considered for the following implementations: Quick system prototyping, Three-tier database systems, Database systems that provide an ODBC driver but no JDBC driver, Low-cost database solution where you already have an ODBC driver.

*Type 2: Java to Native API.* This kind of driver makes use of local native libraries provided by a vendor to communicate directly to the database. This type of driver has many of the same restrictions as the JDBC-ODBC bridge, since it uses native libraries. These libraries must be installed and configured on each machine that will be using the driver. This kind of driver can be considered for the following implementations: as an alternative to using the JDBC-ODBC bridge, since this type of driver performs better than the bridge, or as a low-cost database solution where you are already using a major database system that provides a type 2 driver.



*Type 3: Java to Proprietary Network Protocol* This type of JDBC driver is the most flexible one. This type of driver is generally used in three-tier solutions and can be deployed over the Internet. Type 3 drivers are pure Java and communicate with some type of middle tier via a proprietary network protocol created by the driver vendor. This middle tier will most likely reside on a Web or database server and, in turn, communicates with the database. Using this kind of driver can be considered for the following implementations: web-deployed applets that do not require any preinstallation or configuration of software, secure systems where the database product will be protected behind a middle tier, flexible solutions where there are many different database products in use-the middle-tier software can usually interface to any database product accessed via JDBC, or clients requiring a small “footprint”- the size of the type 3 driver is usually much smaller than all other types.

*Type 4: Java to Native Database Protocol* Type 4 JDBC drivers are pure Java drivers that communicate directly with the database engine via its native protocol. These drivers may be able to be deployed over the Internet, depending on the native communication protocol. The advantage that type 4 drivers have over all the rest is performance; there are no layers of native code or middle-tier software between the client and the database engine. Using this kind of driver can be considered for the following implementations: when high performance is critical, in environments where only one database product is in use, or web-deployed applets, depending upon the capabilities of the driver.

There is a basic flow of steps that all JDBC applications follow.

1. Load the JDBC driver
2. Establish a connection to the database server

3. Execute a SQL statement
4. Process the results
5. Disconnect from the database

The first step in using JDBC is to load the driver. This is usually accomplished using the `forName` static method of the `Class` object (which is part of the base Java system). The call is made as follows:

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

The second step is to establish a connection. JDBC connections are specified by a Uniform Resource Locator (URL), which has the general format:

```
jdbc:<subprotocol>:<subname>
```

where `subprotocol` is the kind of database connectivity being requested (such as ODBC, ORACLE, Informix, etc.) and `subname` provides additional information required to establish a connection. An example of requesting a connection to an ODBC data source named "MyDataSource" via the JDBC-ODBC bridge is:

```
Connection con = DriverManager.getConnection(  
    "jdbc:odbc:MyDataSource");
```

Next step is to create and execute a SQL statement. A `Statement` object has to be created before executing a SQL statement. This is accomplished with a call to the `Connection` class `createStatement` method as follows:

```
Statement stmt = con.createStatement();
```

Then we can execute a SQL statement and return a `ResultSet` object. This is done as follows:

```
ResultSet rs = stmt.executeQuery("select * from students");
```

Then we can iterate the results by making a call to the `next` method of the `ResultSet` object as follows:

```
Boolean more = rs.next();
```

We can use the `getXXX` methods of the `ResultSet` object to get the results of SQL query. The last thing we have to do is to close the connection that we have opened using the `close` method:

```
con.close();
```

This should be called to ensure that the underlying database system frees all the associated resources properly [17].

### 2.3.2. ODBC

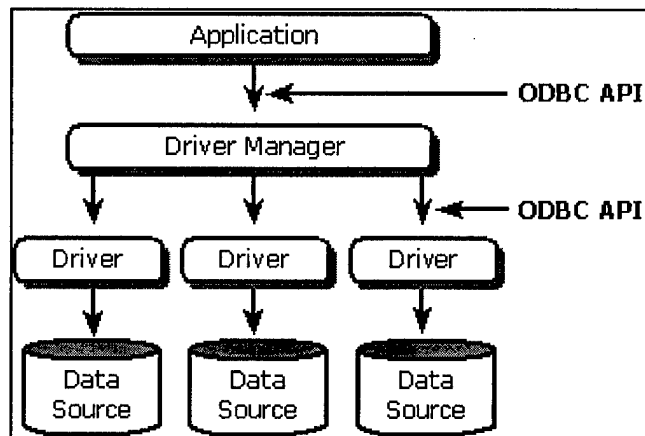
Like JDBC, ODBC is an API for database access. It is also based on the Call-Level Interface specifications from X/Open and ISO/IEC for database APIs and uses SQL as its database access language. ODBC is designed for maximum interoperability—that is, the ability of a single application to access different database management systems (DBMSs) with the same source code. Database applications call functions in the ODBC interface, which are implemented in database-specific modules called *drivers*. The use of drivers isolates applications from database-specific calls in the same way that printer drivers isolate word processing programs from printer-specific commands. Because drivers are loaded at run time, a user only has to add a new driver to access a new DBMS; it is not necessary to recompile or relink the application. [21]

The ODBC architecture has four components:

- **Application.** Performs processing and calls ODBC functions to submit SQL statements and retrieve results.

- **Driver Manager.** Loads and unloads drivers on behalf of an application. Processes ODBC function calls or passes them to a driver.
- **Driver.** Processes ODBC function calls, submits SQL requests to a specific data source, and returns results to the application. If necessary, the driver modifies an application's request so that the request conforms to syntax supported by the associated DBMS.
- **Data source.** Consists of the data the user wants to access and its associated operating system, DBMS, and network platform (if any) used to access the DBMS.

The illustration in Figure 2.3 shows the relationship between these four components.



**Figure 2.2: The ODBC Architecture**

There can be multiple drivers and data sources. This allows an application to simultaneously access data from more than one data source. The ODBC API is used in two places: between the application and the Driver Manager, and between the Driver Manager and each driver. The interface between the Driver Manager and the drivers is sometimes referred to as the *service provider interface*, or *SPI*. For ODBC, the API and the

service provider interface (SPI) are the same; that is, the Driver Manager and each driver have the same interface to the same functions [21].

## ***2.4. Previous Work***

There are a few theses previously done in AFIT about web based application development and integrating relational databases and other tools in a web based environment. One of these is done by Daniel L. DiPiro [2]. In his thesis DiPiro focuses on “a methodology that can be used in the analysis of a web based data access enterprise to aid the developer in choosing the right software technologies.” After giving some information on the client-server model, current Internet technologies, and some database access techniques he defines his methodology. His methodology “examines the key environmental factors affecting the design of web software components and the areas in which they should influence design decisions.” He also gives an example implementation of his methodology as a case study.

His methodology has three main steps:

1. Development Environment Analysis
2. Component Analysis and Design
3. Function Implementation

Each of these steps has some other sub steps.

The development environment analysis is divided into three components. These are client environment, existing data source architecture, and existing resource environment. He does his client environment analysis according to the number of potential internal and external clients, and the hardware and software platforms they use. He differentiates the clients as internal and external according to their network connectivity.

Then he does his analysis of existing data source architectures. This is important when deciding on the web-based database access technology for the system. This analysis includes existing database management systems (DBMSs) and the location of data sources.

The third phase of the development environment analysis is the analysis of the existing resource environment. This includes time, money, and personnel. The availability of these three resources is very important in a development effort. The designer will generally be limited in one or two of these.

The second step of his methodology is component analysis and design. This step of the methodology includes functional analysis and functional decomposition of the application. While doing the analysis of the components, generally Data Flow Diagrams (DFD) are used.

The third step, function implementation, consists of analysis of possible technologies that can be used to implement each sub-component process, designed in the component analysis and design step. This analysis is done in terms of how each technology can interface with other technologies. He gives a small description of each technology and some implications of choosing that technology to implement a sub-process.

In the last part of his thesis, DiPiro implements his methodology in a case study. In this case study, he designs an application to provide access to existing AFIT/CI data sources.

Another thesis on a similar topic was done by Tien-Chen Lee (Taiwan AF) [1]. He designs a web-based prototype for AFIT Edplan Administration in his thesis. He uses

DiPiro's methodology while analyzing and designing the web-based prototype. Lee's thesis work is an implementation of DiPiro's methodology.

After applying the methodology, he chooses Microsoft technologies for his system. He chooses ASP technology for designing web pages, ODBC for the database connectivity, and Microsoft Access as the database. The overall system he designed has some limitations. These limitations result from sticking to only one technology while designing the user interface and web pages. He only uses pure HTML while designing the user interface. This limits the ease of use of the interface. It could be done in a user-friendlier manner by using ActiveX components and scripting languages on the client side. As he states in the conclusion of his thesis, another drawback of his system is that it does not support concurrent access to the database. This is because he uses Microsoft Access as the database management system. A more powerful RDBMS, like Oracle, which is available in AFIT, could be used as the DBMS.

Lee finds DiPiro's methodology very helpful to the developers while making decisions for choosing the right technologies to create web-base applications. He also finds it helpful during the implementation of the design. He states that DiPiro's methodology narrows down the range of technologies to be chosen from. This reduces the time needed for creation of web-based applications. The overall comment of Lee on DiPiro's methodology is that it was complete and thorough for the design and implementation of the web-based AFIT Edplan administration application that he created.

Another thesis on integrating software tools is done by Penelope Noe [3]. In her thesis, Noe defines a methodology for tool integration in a structured approach. Her integration methodology is based on the concept of a design space, composed of functional

and structural dimensions. The functional dimensions of the design space identify the requirements for the tool that most affect the solution for the integration effort. The structural dimensions of the design space determine the overall framework of the integrated system. The characteristics of the tool pair being integrated are defined in the functional dimensions and the characteristics of the resulting integrated system are defined in the structural dimensions in her methodology. She also defines some design rules, which are guidelines for choosing between structural dimensions given a set of functional dimensions, to do the mapping from the functional dimensions to the structural dimensions.

She defines two sets of functional dimensions: one for a single tool and one for the tool pair. The functional dimensions for a single tool are input characteristics, output characteristics, and tool extendibility. The functional dimensions for the tool pair are extendibility class and data compatibility.

The structural dimensions for the system include the communication path, control integration implementation, and data transformation. Then she defines some design rules, which are used to decide which method of integration to choose. Design rules use the values given to each of the functional dimensions, for both a single tool and the tool pair, to determine the structural dimensions for the integration. These design rules are applied to determine the structure of the system.

Then she uses this methodology to integrate different software with AFITtool. She integrates Rational Rose 98, a CASE tool for software development, Acme parser, and daVinci, a graph layout tool with AFITtool.



This integration methodology considers only data and control integration of Wasserman's five integration classes: platform, presentation, data, control, and process integration [8]. She explains this as due to the nature of the integration effort.

## ***2.5. Summary***

In this chapter, some of the web-based technologies are described briefly. These technologies are separated as server side (CGI, Java servlets, JSP, ASP) and client side (Java applets, ActiveX controls, DHTML) technologies. A general architecture of a web-based application is illustrated. This way we will have a better understanding of the web-based technologies used in this research. The two widely used database connectivity methods (JDBC and ODBC) are also mentioned. In the last part of the chapter a brief summary of some of the related research done previously is given.

### **3. Web-based Integration Methodology**

Integrating tools or applications in a web-based environment is a similar task to tool integration in general. But of course there are some differences. Generally the overall purpose is to form a complete system using the available tools. Another goal may be making the system available from outside of the organization. This might be necessary when a company decides to make some of its resources available via the Internet. This kind of integration may also be needed when two companies merge. Each of them may need to use tools or applications of the other company or a combination of applications or tools from both of the companies. The merging companies might need to integrate their databases over the Internet in some way.

#### ***3.1. Overview***

This integration methodology is based on the Internet platform that generally consists of a web server, an application server, a database server, and client browsers. Before starting to explain the methodology we have to identify the properties of the Internet platform, and draw a general picture of it. Then we apply the methodology steps shown in Table 3.1. In order to have a well-integrated system first we have to perform a good analysis of the tools at hand. This analysis will include the extendibility and the source code if available, the input and output characteristics of the tools, and the user interfaces.

1. Analyze the tools
2. Choose an integration method
3. Analyze the target platform
4. Design the system
5. Implement the design

**Table 3.1: Steps of the integration methodology.**

The second step is choice of an integration method according to the results of analysis of the tools. While choosing the integration method we basically use Noe's approach [3]. The third step is the analysis of the target platform in which we will list the properties of the available resources. After the analysis of the target platform we can start designing the integrated system. During this design phase we have to make use of some basic software engineering techniques. After finishing the design of the system we have to implement it. DiPiro's methodology will help us while implementing our design [2].

### ***3.2. Analysis Of The Tools***

Analysis of the tools to be integrated will include what Noe describes as the functional dimensions of the tool set. While doing this analysis we will characterize the tools both individually and as a pair like Noe does in her integration methodology. This will be the first level analysis of the tools. After making some design decisions and choosing the integration method in the following steps of the methodology we may need to make some detailed analysis of the tools. The first level analysis will include three steps. We will first analyze the extendibility and availability of source code of the tools. Then we will analyze the input and the output characteristics of the tools. We also need to analyze the user interfaces.

### *Extendibility and Source Code Availability*

Extendibility of a tool is very important for the integration process. Extendibility of a tool means that it is possible to extend the tool beyond its current capabilities. The integrator can add the necessary functions and methods to the tool for the integration purpose. [3]

Availability of the source code is very important for the extendibility of a tool. If the source code is available the design of the integrated system can be easier. We can separate the code so that some of the code runs on the client side and some of it runs on the server side. We can recompile the tool to run on a different hardware or software platform. We can add new methods and functions to ease the integration process or to add new capabilities to the overall system. We can make some changes to the tool to be able to integrate it.

In most cases the source code of the tool is not available. But generally there are other ways of extending the tool. Some tools supply a scripting language and capability to add new menu items and options to the tool. The user can add new menu items and associate them with written scripts to extend the tool or to automate some process sequences [3]. Others may supply macro capabilities for the same kind of extendibility. There may be other proprietary extendibility methods for the tools. After making the analysis we can characterize the tool with one of the four alternatives below:

- Source code available
- Provides an extendibility method
- Both of these
- Not extendable

When we finish the extendibility analysis of both of the tools we will come up with one of three possibilities. These are:

- None of the tools are extendable
- One of them is extendable
- Both of the tools are extendable

### *Input and Output Characteristics*

This part of the analysis will give us the methods that the tools use to obtain their data and to output the results of the functions they perform. The input and output systems can be either a persistent data store or a non-persistent data store, or a combination of them. Noe lists four input methods that a tool may use:

*Standard Input:* This is the default input mechanism for many applications. Standard input supports operating system redirection. If the tool is a command line application everything entered from the keyboard after the tool is executed is referred to as stdin.

*File:* The data needed by the tool may be held in one or more files in the file system or in tables within a relational database management system. These files may be created by the user according to some file format specifications or may be created at installation time. The path and the name of the files can be fixed and internal to the tool, may be supplied by the user at run-time, or may be held in environment variables. For the purpose of this integration methodology a database will be considered as file input mechanism.

*Command Line Parameters:* Command line parameters are given with the execution command for the tool. Very little input can be given with command line pa-

rameters. Command line parameters are mostly used for giving some directions to the tool.

*Message Passing:* A tool that uses message passing for data input expects data and control messages in a special format. To perform its functions it has to receive these messages. It is possible to capture the messages to and from the tool to control the system and monitor the flow of communication.”

*GUI:* This is another way of data input mechanism for a tool that is not considered in Noe’s methodology. Since we are doing a web-based integration the GUI must be considered as an input mechanism. A GUI is formed by using some kind of windowing system like Windows, X-Windows, Java’s AWT or Swing, etc. When a GUI is used as an input mechanism the data is entered in the tool by using some graphical forms and some menu system.

Noe lists four alternatives for the output mechanism. These are:

*“Standard Output:* This is similar to standard input. Stdout is the default output mechanism for many applications. Just like stdin, stdout has support for operating system redirection.

*File:* The results are written to an output file with a specific format. The name and location of the file can be internal to the tool, can be given at run-time by the user, or can be held in environment variables. As we have done for the file input mechanism we are going to consider a database system as a file output mechanism.

*Message Passing:* Tools that use message passing as an output mechanism send formatted messages containing the result data to external destinations.

*Built-in Output:* This includes the printer, standard error (stderr), and log files. The printer is one of the most used output mechanisms. Standard error is the default output mechanism for the output of errors. The log files are used for writing output messages about the events that occur during the execution of the tool.”

*GUI:* Here again we are going to consider a GUI as an output mechanism also. When the GUI output is used the output of the tool is presented within the GUI, probably in a separate window. This is generally for seeing the results of the tool graphically on the screen. Generally it is possible to save the GUI output to a file.

### *User Interface*

The analysis of the user interface is important for presentation integration purposes. The user interface of the integrated system should be similar to the original user interfaces of the tools. The new interface should give the same or better functionality. It should get rid of the drawbacks of the old interface. It should give new functionalities.

We have three possibilities for the user interface of the tools:

- The tool has a graphical user interface (GUI).
- The tool has an interactive interface.
- The tool does not have a user interface and can be run by a command with command line parameters.

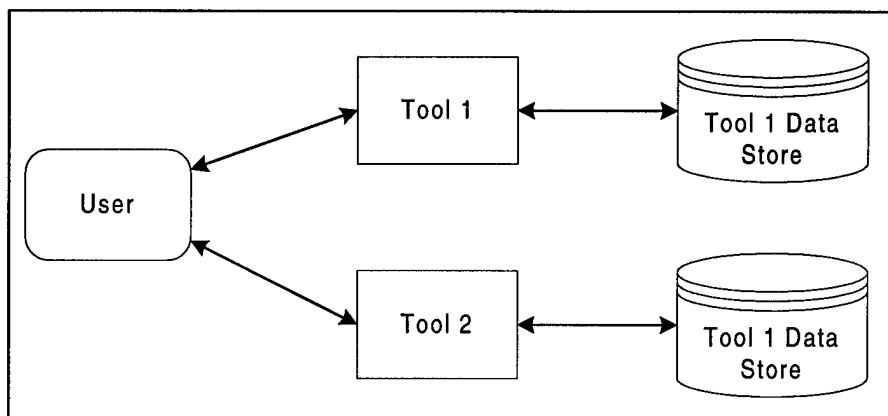
If both tools have a GUI and if the functionality of the tool is in the GUI it will be very hard to integrate these tools. It may even be impossible to integrate them.

If one of the tools has a GUI and the other has a command line interface integration can be easier. Depending on the properties and capabilities of the tool we can choose a way of integration.

If neither of the tools has a GUI we may need to design a new interface for our integrated system. This new user interface can be a GUI in a browser or plain HTML based interface, which includes a form or a series of forms. This part will depend on the choice of technology for the web interface and our design decisions

### **3.3. Integration Methods**

Before the integration we have two or more tools, each of which performs a different task that is part of an overall goal or helps us to reach the overall goal. But the user has to run each tool separately, probably inputting the same information in different formats to each tool. The output from each tool is taken separately in different formats. The tools may need to be run in an ordered sequence, so that the output of one will be used as an input to the other. The interaction of the user with the tools before the integration process can be seen in Figure 3.1.

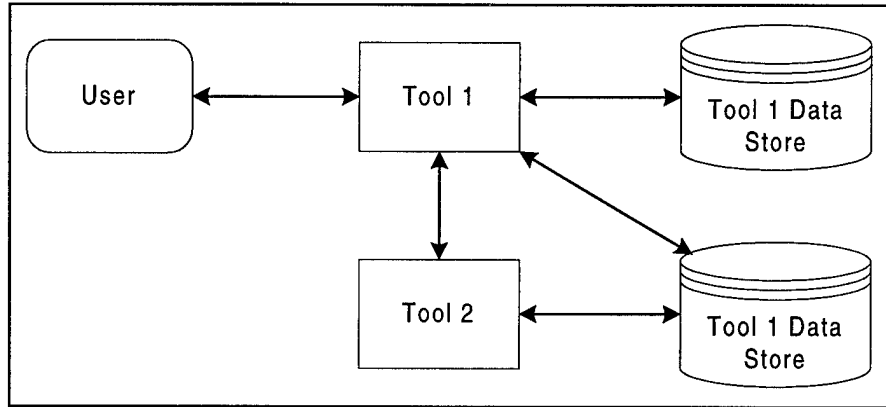


**Figure 3.1: Interaction of the user with the tools before integration.**

Our goal is to have a well-integrated system that can be run in a web-based environment. The degree of integration will depend on the properties of the tools. Thomas and Nejme define the integration as “Integration means that things function as members



of a coherent whole.” [9] We have to integrate the tools in a system such that the system serves our overall goal as a whole.



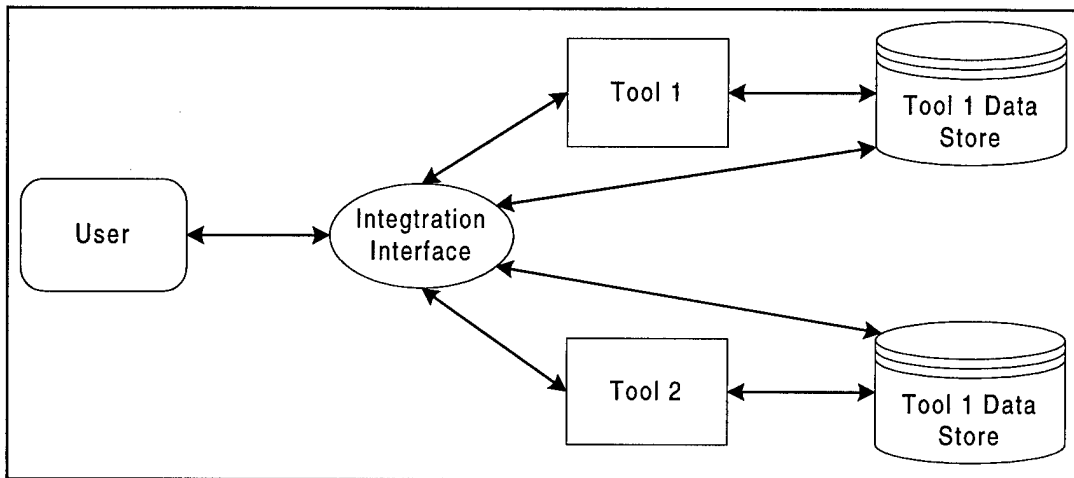
**Figure 3.2: The first integration paradigm.**

Now we have to define some integration frameworks while integrating tools in the web based environment. We can suggest three different integration paradigms from the perspective of presentation and control integration at this point. Choosing between these paradigms will mostly depend on the properties of the tools that we have learned during the tool analysis phase.

The first paradigm, shown in Figure 3.2 is to use the first tool to control the system and to run the second tool from the first tool. This can be done in many different ways that will depend on the detailed properties of both of the tools. In this paradigm the user will interact with the system via the first tool. The first tool will be extended to include functions and methods to supply the input data for the second tool, send the data to it, run the tool with the input data, make the necessary communication between the tools, and get back the results from the second tool and present the results to the user.

The second paradigm, shown in Figure 3.3, is to design and create a new interface to the tools. This interface is a front end between the tools and the user. It gets all the

necessary input from the user, makes any necessary transformations, sends the inputs to the tools, gets the outputs and presents them to the user in the desired format. This interface runs the tools in the order that they have to be run, and makes the necessary communication between the user and tools and between the tools. This user interface might also directly access the data stores that the tools use making it possible to access the available data in the data stores directly from the integration interface.



**Figure 3.3: The second paradigm**

The third paradigm, shown in Figure 3.4, is slightly different from the second paradigm. In the third paradigm we still have to design an integration interface. But the function of the integration interface is a little different. The integration interface is responsible for the control and presentation integration of the tools. The interface gets the necessary input from the user and sends this input to the tools. Then it gets the output from the tools and presents them to the user. In this case the integration interface doesn't need to do any data transformation since there isn't a data compatibility problem here. The tool can read the data of the other tool without the need of any data transformation process.

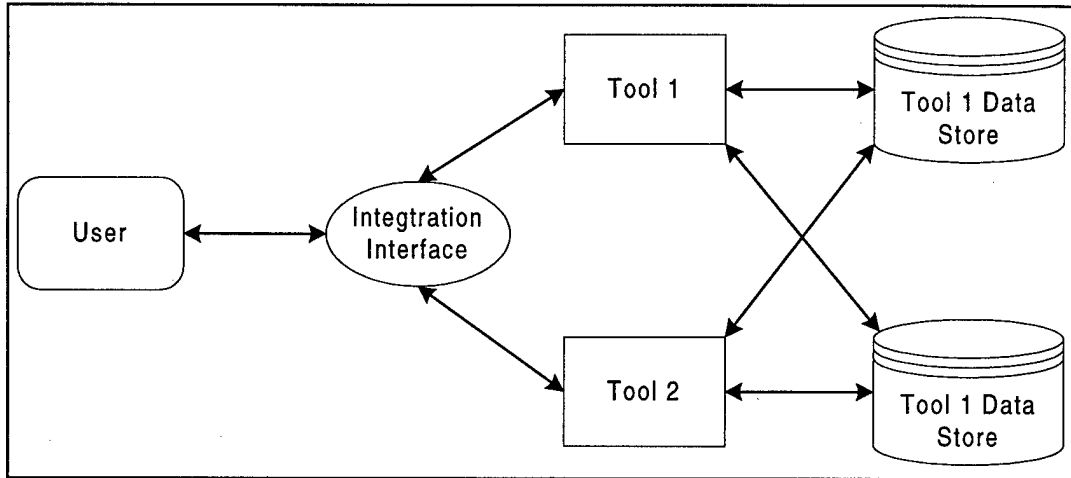


Figure 3.4: The third paradigm

### 3.4. Choosing An Integration Method

	First paradigm	Second paradigm	Third paradigm
None extendable		✓	✓
One extendable	✓	✓	✓
Both extendable	✓	✓	✓

Table 3.2: Choosing the integration paradigm

While deciding on the integration paradigm that we are going to use we need to consider the results of the tool analysis. While choosing the integration method we basically use Noe's approach. We have to consider each of the three possibilities of the extendibility analysis. As we can see in Table 3.2 we can choose all the paradigms for all the extendibility options except the first paradigm for non extendable. We also have to take into account the data communication and the data transformation between the tools.

#### *None of the tools are extendable*

If none of the tools are extendable we cannot control the system using one of the tools. We have to add an interface to the system that will take care of the control of the

tools. In such a case we have to choose the second or the third paradigm. Here we look at the compatibility of the input/output data of the tools. If the data is not compatible then we will need to add some data transformation routines, which will lead us to the second paradigm. If the data is compatible then we won't need any data transformation routines. In this case we can choose the third paradigm.

As I have explained above we have to design an integration interface between the user and the tools. This integration interface will be a middleware application. Since we have to design this middleware application from scratch we can design it as a distributed application. We can design it so that some part of it runs on the client side and some part of it runs on the server side. This middleware application has to get the necessary input from the user and pass it to the first tool or write the input to the shared database if the tool can read the input from the shared data. Then the middleware application runs the first tool with the inputs. If there is a data transformation needed between the two tools during the data exchange the middleware application does this job also. The middleware application also gets the input for the second tool from the user. Then the middleware application runs the second tool. Then it presents the output of both of the tools to the user. As stated by Noe "the nature of the tools dictates the implementation chosen" for the middleware application.

#### *One of the tools is extendable*

If only one of the tools can be extended then we can choose any one of the integration paradigms. Our choice will depend on the way the tool is extendable and other characteristics of the both of the tools. There are three possibilities as explained in Section 3.2; source code is available, the tool provides an extendibility method, or both.

If the source code of the tool is available it will be better if we choose the first paradigm. In this case we have to extend the tool by adding necessary methods and functions to the tool to make it possible to control the second tool from the first tool and to make the data transformation between the tools. We can also choose the second or the third paradigm. There can be several reasons to this. For example it may be easier to design a new interface to the system than playing with the source code of the tool. The integrator may not be familiar with the programming language used to code the tool, or it may be impossible to extend the tool in order to control the second tool. So, instead of trying to learn the new programming language the integrator may choose the second or the third paradigm and design an integration interface and implement it with a programming language that he knows best.

	First	Second	Third
Source code	✓		
Provides a method		✓	✓
Both	✓	✓	✓

**Table 3.3: Choice of integration paradigm according to the extendibility method**

If the source code is not available but the tool provides extendibility in some way it will be better if we choose the second or the third paradigm. Then we can design and implement the integration interface between the tools and the user, which can be distributed between the client and the server computers. We can either add the data transformation functions to the integration interface, or extend the tool to take care of this if it is possible.

If the tool provides both of the extendibility options then we have more flexibility on choosing the integration paradigm. We can choose the one that best fits the specific

needs of our system. We have to take into account our requirements for the integration. In Table 3.3 we can see the applicable integration paradigms for the extendibility ways provided by the tool.

#### *Both of the tools are extendable*

This case gives us more flexibility for the design of the integrated system. But we have to check the extendibility ways of the tools. There will be nine combinations for the extendibility ways of the tools. We can choose any of the paradigms for all these combinations. If both of the tools are extendable through source code availability then we should choose the first paradigm. If none of the tools' source code is available we should choose the second or the third paradigm. For the rest of the combinations the choice will be up to the integrator. The integrator can choose the one that will be easiest according to him.

For example if the source code for one of the tools is available and it is implemented in Java then we can choose the first paradigm. Then we can separate the tool so that the GUI of the tool that accepts the data input from the user is run on the client side and other routines that process the data are run on the server side. We add new modules that will control the second tool and make the data transformation for the second tool to the server side. We can distribute the workload between different machines by running the second tool on another computer. We can extend the second tool to take care of the data transformation between the tools and not increase the work done by the first tool.

#### *Data Communication*

There will be a data exchange between the tools. There are two systems for data exchange between the tools as described by Noe. These are sharing the same data and passing the data between the tools.

Sharing data: The data sharing mechanism may be files, common objects, or databases. Generally many tools share data by reading from or writing to the same files, using common objects, or accessing the same databases. Making the data exchange between the tools using shared data may have some problems, which have to be solved during the design of the system. These problems may include resource locking, stale data, and timing problems that can be caused by synchronous data access.

Passing data: The data needed by one tool can be sent from the other tool. The tool can send the data in one of the three ways: stdin/stdout, message passing, or middleware.

#### *Data Transformation*

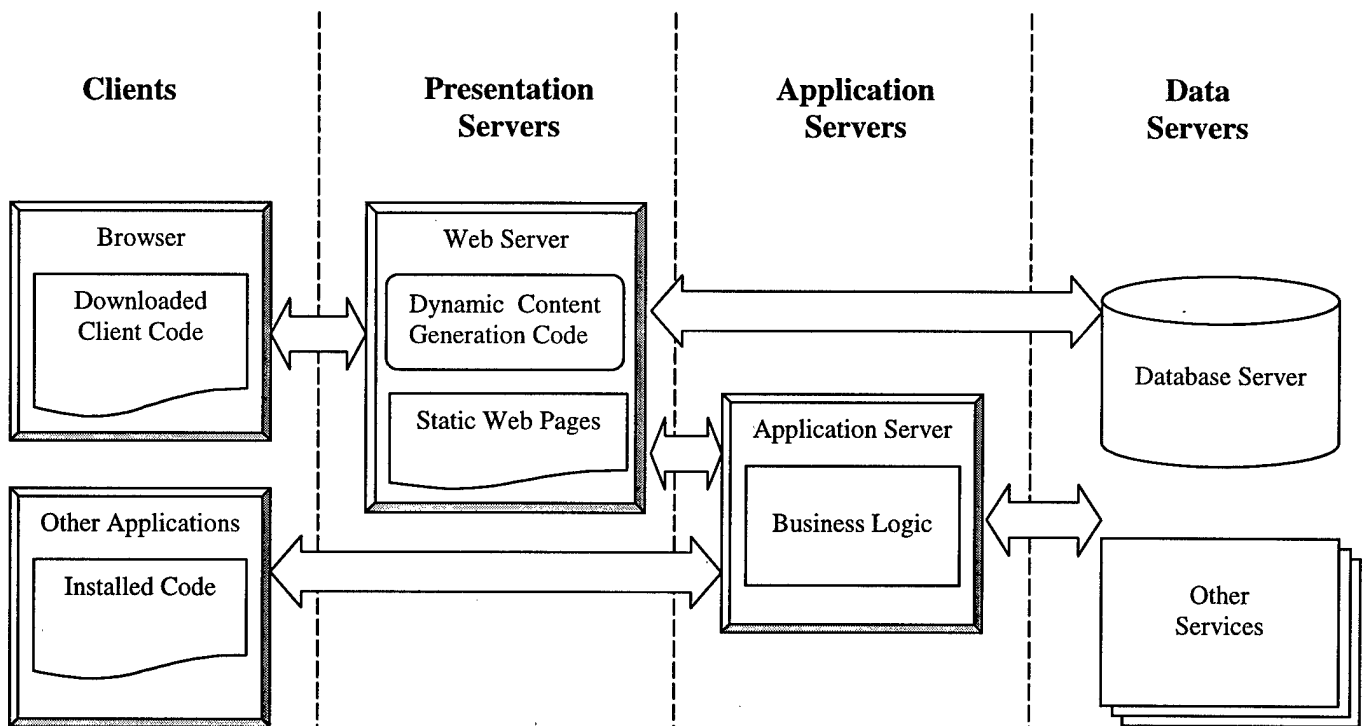
The data that the tools pass to each other has to be in the same format. If the data that the tools use is not compatible than we have to pass the data through a transformation process. Compatibility here corresponds to the format of the data. The data transformation can be done in different ways. A tool can transform its output to make it compatible with the input data for the other tool. The tool that needs the input from the other tool can transform the data while inputting it. Both of the tools might need to transform the data during input and output. These three choices can be used if the tools can be extended to transform the data during input and output. If the tools are not extendable the data transformation must be done by an external source.

### **3.5. Analysis Of The Target Platform**

The philosophy behind the Internet is to share information, and provide universal access through a web browser. The Internet computing model distributes information

simply to any user without the complexity of the client-server computing model. With Internet computing, all information is in one place to be accessed, used, and managed.

We have to make a definition of the Internet platform before starting the integration methodology. The basic requirement for the Internet platform is “anytime, anywhere” access to the applications. All of the clients must have access to the applications any time and from anywhere. This requirement will limit the presentation layer of the application. The Internet platform generally has a multi-tier architecture. You can see a generic Internet platform in Figure 3.5. [31]



**Figure 3.5: A Generic Internet Platform**

“The client tier is generally a browser-based HTML client. This can support the “any time, anywhere” requirement of the Internet platform, which is the basic requirement for it. Complex GUIs can be supported within the browser by downloading more client-side application code, relying on browser plug-ins, or



relying on browser-specific scripting capabilities. Another way for satisfying complex GUI requirements is installing the application code on the client computer and running outside the browser. This way we can eliminate the security restrictions of the browser. But this is not the desired way because the user has to install new software on his computer.

The presentation server tier consists of web servers. The web servers can hold the static HTML based web pages to be downloaded by the clients. The web server also handles HTTP requests from the clients and dynamically generates presentation code for execution and display on the client side, typically in the form of HTML pages. The HTTP requests may require connections to other servers like application or data servers, or the use of other server-side resources.

The reusable business logic is kept inside the application servers. The business logic can be reused across a variety of clients and applications. The reusable components inside the application server should be easily callable from external servers and clients to facilitate application integration requirements. The application server layer can also contain specialized reporting and analysis servers to handle high-end business intelligence requirements.

The data server tier contains databases and other resources that must be integrated into the system. The data servers should be scalable enough to meet the intense data storage, manipulation, retrieval, and analysis requirements of Internet applications.” [31]

Security is generally a very big concern in the Internet platform. Clients that are outside the organization have to access the servers through a firewall. That is, the presen-

tation, application, and database servers are behind this firewall. The firewall limits the access to the servers by allowing the use of only certain protocols (like HTTP, FTP). Firewalls also limit the visibility of the servers to the outside world. In most cases application servers and data servers are not directly visible to most of the clients. Because of the firewall the clients can access the data servers only through the web servers and application servers, or in some cases only through the web servers by using only certain protocols.

A very important thing in the web-based integration of tools is to decide on the platform on which you will run the tools. This can be either the client side or the server side. There are a number of things that will affect this decision.

- The requirements of the tool: If we want to run the tool or tools on the client side all the clients have to be able to run the tools. All of the clients must be able to meet the hardware and software requirements for the tools.
- The number of users: The number of users of the new system will have an important effect on our decision of choosing the platform. If the number of users is very high choosing the client computer to run the tool will not be a good option. Because in this case we will need to install the tool on all of the client computers. But if the number of users is very small than this may be a good option. This way we can eliminate the need for design of a new interface.
- Availability of the data stores to the clients side: The data stores that the tools use have to be available to the client side. In other words the tool must be able to access its data stores if it is run on the client side. This can be af-

ected by the security policies applied to the network. Access to the data server that the tool uses from some of the clients may be restricted.

- The effects of the communication protocols between the clients and the server: There may be some restrictions on the communication protocols between the client and the server. Only some specific protocols might be available (like HTTP and/or FTP). We might need to wrap the communication protocol of the tool within one of the available protocols and this might be inapplicable for our tools.

If we choose the client side to run one or both of the tools we have two options: We can run the tools within a browser or as separate applications. In order to run the tool within the browser the tool must be a java applet or an ActiveX component. The tool that will run in the browser will be downloaded from the server within an HTML page, so we won't explicitly need to install the tool on the client computer. If we want to run the tool as a separate application we have to install it on every client computer.

If we want to run the tools on the server side we will need a means of access to the tool or tools from the client. We can do this by means of an interface application. This interface application will have a server side component and a client side component. The client side component will get the input necessary for both of the tools and send them to the server side component. The client side will also supply the Graphical User Interface if a GUI is needed. The server side component will get the input from the client side component and run the tools with this input. Then it will get the output from the tools and sent it back to the client.

As part of the initial analysis of the target platform we first have to list the following resources and then make an analysis of them:

- Available web servers
- Firewalls and/or proxy servers
- Application servers

The analysis of the available web servers will include their properties, capabilities, requirements, and supported platforms. We have to gain information about the server side technologies supported by the web server. The analysis of the firewall will include the general architecture of the system, and the protocols that are let through the firewall. Then we are going to analyze the application servers available if any. This analysis will include the architecture of the application server and the supported technologies.

The results of this analysis will help us for the design and architecture of our system, and choice of target platform for running the tools.

### ***3.6. Design of the System***

In the design phase we will create a system design that will be based on our choice of integration method. Basically we can choose one of the three integration paradigms that were described above. But we can create different designs from these paradigms. The design will again depend on the results of the analyses of the tools and the analysis of the target platform. Having the choice of integration paradigm and some initial design decisions at hand we first have to do some detailed analysis of the tools. We will do the detailed analysis according to the choice of integration paradigm. Generally we will do a two level system design:

- Design a front end (Presentation level)

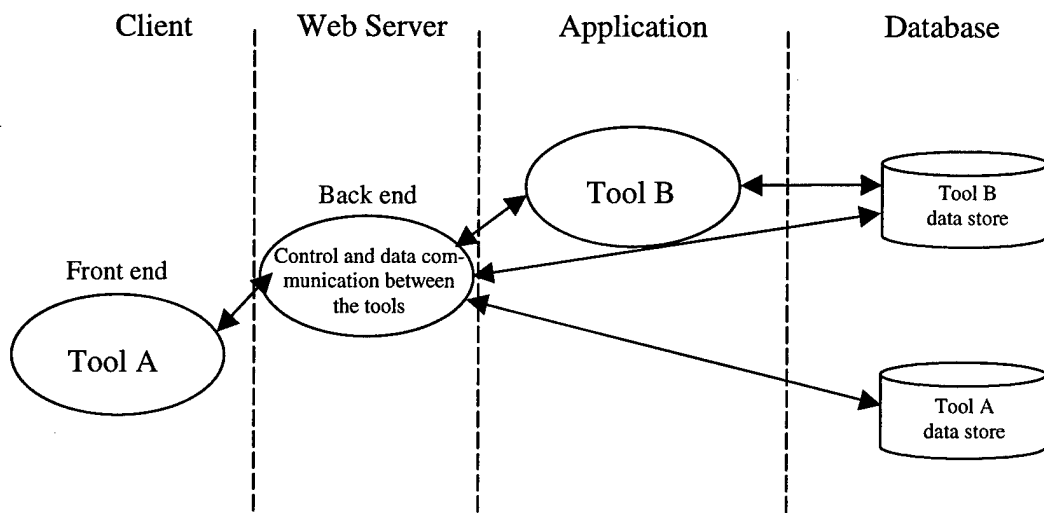
- Design a back end (Integration logic level)

The front end will basically include the user interface of the system, which will be a GUI on the client side or an HTML based interface. The back end will be on the server side and will include the actual integration logic of our system. The integration method will be our starting point while doing the design of the system. After doing a more detailed analysis on some parts of the system according to the integration paradigm we will design the front and the back end of our system.

*The first paradigm*

If we have chosen the first paradigm as the integration method, the design of the front end and the back end of the system will depend on the extendibility way of the tool that we are going to extend.

We have different options for choosing the platform on which we are going to run this tool. We can run the first tool completely on the client side, completely on the server side, or distribute the tool between the client and the server. This system design can be represented as seen in Figure 3.6.

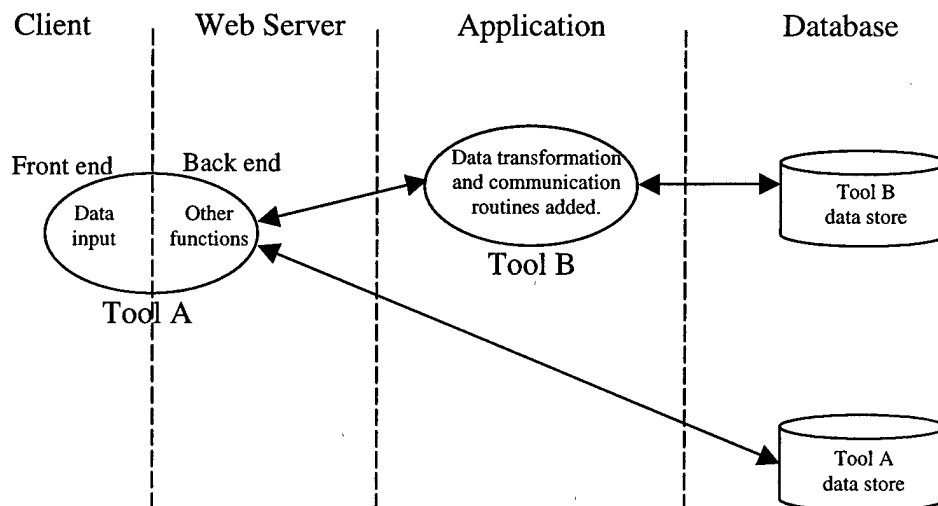


**Figure 3.6: System design using the first paradigm and Tool A as front end.**

We can run the tool on the client side only if it is possible to migrate it to the Internet platform. We can do this by running the tool within the browser. This is possible if the tool is programmed in the Java programming language as an applet or programmed as ActiveX components. Since not all browsers have support for ActiveX components we should choose the client side for running the tool only if the tool is implemented as a Java applet. But in this case another issue is the location of the data source and size of the data that the tool accesses. The database server must be visible to the client so that the tool can access the data source, and the size of the data passed back and forth between the tool and the data source shouldn't be very large because of the bandwidth restrictions of the connection between the client and the server. We can also route the communication between the tool and the data source through the web server. We can do this by wrapping the communication protocol between the tool and the data source in HTTP protocol and adding some functions to the web server that can pass the data request to the database server and pass the responses to the client. At this point we have to make a detailed analysis of the tool in order to be able to make decisions and design the system. We can do this from the following aspects:

- Analyses of the data flow within the tool. We can do this by using classical data flow diagrams (DFD)
- Analysis of the input and the output mechanisms
- Analysis of the database access
- Analysis of the source code according to the security restrictions of the integrated system under consideration

Another option may be distributing the functions of the tool between the client computer and the server. But we have to figure this out according to the analysis of the source code of the tool. We can run the user interface of the tool on the client computer and other functions and integration logic on the server computer. For example it is desirable that the data input and data validation functions are on the client side and data processing functions are on the server side. We have to add new methods that will run on the server side and make the necessary data transformations for the data exchange between the tools. We also have to add necessary methods that will invoke the second tool with the data input, get the output of the second tool, and send the output to the client. The system design will look as in Figure 3.7 when the first paradigm is used and the functions of the extended tool are distributed between the client and the server. Here the user interface on the client side is the front end, and other functions that reside on the server side are the back end of the system.



**Figure 3.7: System design using the first paradigm and Tool A is distributed between the client and the server.**

If the source code of the tool is not available or after making a detailed analysis of the source code we decide that it is impossible to run it on the client or distribute and run it on the client and the server computers then our system design will look as in Figure 3.8. In this case the front end and the back end of the system must be designed from scratch. While designing these parts of the system we have to consider the target platform. We have to design our system according to the available resources or we have to be able to supply the requirements of the system that we have designed. The front end of this system can be a GUI, which can be done as an applet or ActiveX component, or form-based HTML pages. While designing the front end of the system it will be good if we try to make the look and feel of the user interface as the original user interface of the tool. The back end of the system will be on the web server and get the data input from the client side and pass the data to the tool. If the tool has a GUI and data is normally input through this GUI we can design an applet on the client side and make it look like the original GUI. Then we have to design the back end of the system, which will be on the

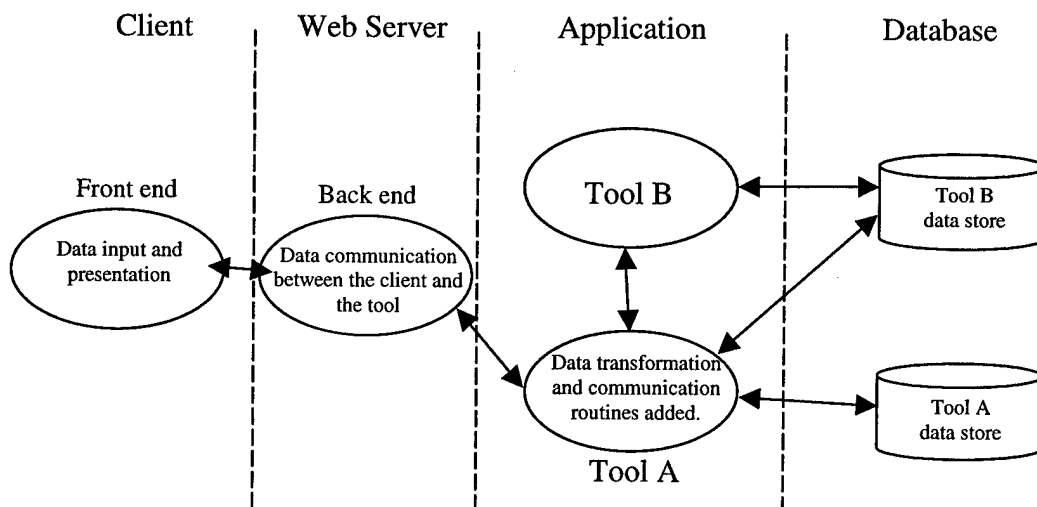


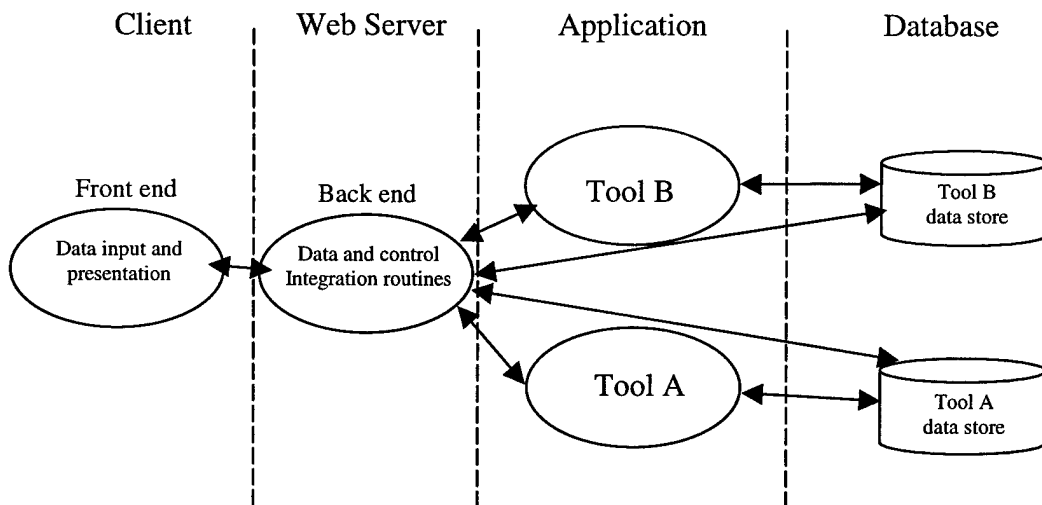
Figure 3.8: System design using the first paradigm with a newly designed front and back end.



server side. This back end of the system will take the data input from the front end of the system and pass it to the tool, and get the results and output from the tool and pass them to the front end of the system.

*The second paradigm*

If we are doing a design for the second paradigm then we must design a front and a back end for our system. Our system design will be as seen in Figure 3.9. The front end of the system will be the same as described above for the first paradigm. The back end of the system will include the functions that will get the input data from the front and pass it to the back end and get the output from the tools and pass them to the front end. The integration logic will also be on the back end. It will include the data communication and data transformation functions between the tools.



**Figure 3.9: System design using the second paradigm.**

*The third paradigm*

If we are designing a system for the third paradigm our back end design will be different. The back end of our system will include the data communication routines between the front end of the system and the tools. It will get the data input from the front end and pass it to the tools, and get the output from the tools and pass it to the front end.

Since there isn't any need to do data transformation there aren't any data transformation functions on the back end system. The system design for the third paradigm will be as seen in Figure 3.10. The front end of this design will be the same as the one for the second paradigm. It will get the data input from the user and send the data to the back end of the system.

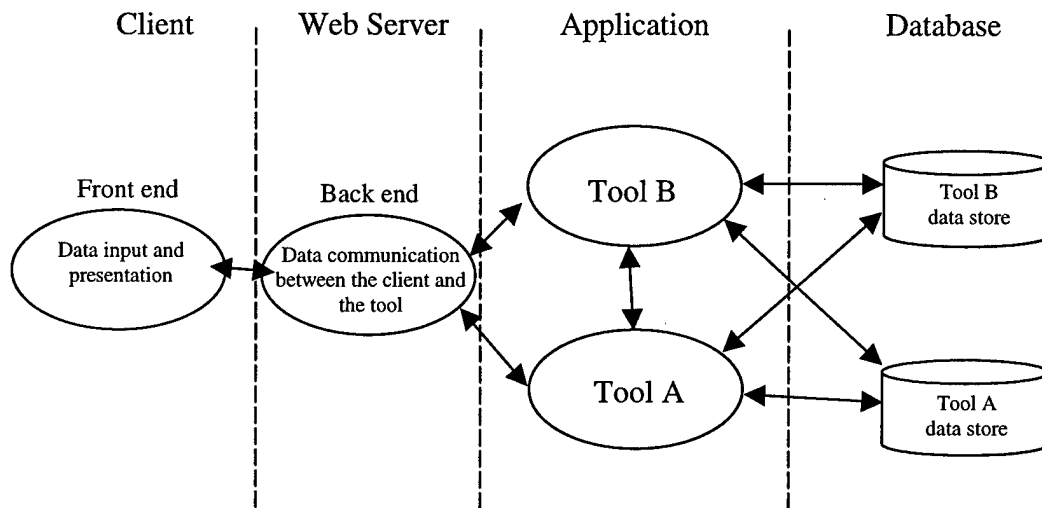


Figure 3.10: System design using the third paradigm.

### 3.7. Summary

In this chapter a methodology for integrating tools on a web-based environment developed as part of this research effort is presented. Based on the definition of a web-based environment the integration of the tools is can be done in five steps. These five steps are: 1. Tool analysis, 2. Choice of Integration Method, 3. Target Platform Analysis, 4. System Design, and 5. Implementation. An example application of this methodology is presented in the next chapter.

## **4. Application of Methodology**

In Chapter 3 a methodology is defined to integrate tools in a web-based environment. In this chapter an application of this methodology will be demonstrated on some existing tools in AFIT. These tools are part of the Edplan creation and administration system of AFIT. The first tool is the Edplan administration tool. This tool was created by Jason Gunsch as a summer project. This tool basically helps the students to create their Edplans and save them in a database. The tool also has a part for the database administrator to enter and edit the information about the courses, course requirements, advisors, and students. The second tool is the Edplan checker tool. This tool checks the Edplans created by the students and reports if they comply with the requirements of AFIT, and if the students meet the prerequisites and co-requisites of the courses they want to take. Our purpose is to integrate these tools so that the students can create the Edplan, check it, and save it on a web-based environment.

### ***4.1. Analysis Of The Tools***

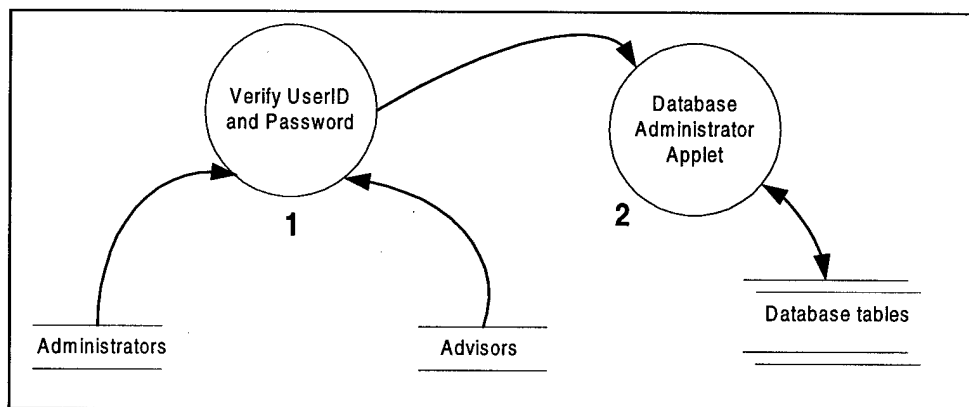
In this step we characterize the tools first individually, and then as a pair. The first tool that we are going to integrate, the Edplan Administrator, is written in the Java programming language and its source code is available. We characterize the extendibility of this tool as “source code available.” This tool does not supply any other way of extendibility. After making a low level analysis of the source code of the tool the following properties of the tool were discovered:

- The tool is designed as Java applets. There are two Java applets; one of them is for the database administrator and the advisors to edit and admin-

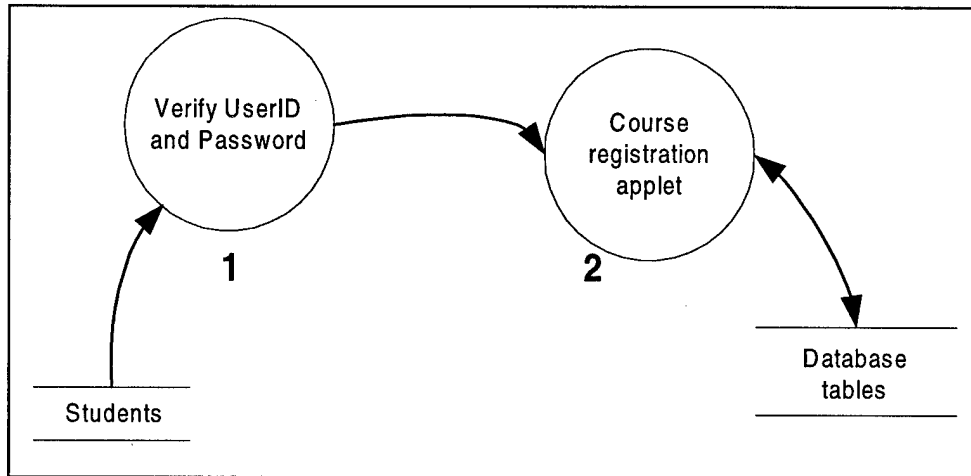
ister the course registration database and the other is for the students to create their Edplans and save them.

- The tool has a GUI and this GUI is programmed using Java's AWT (Abstract Window Toolkit). All the control of the tool and the user data input to the tool is through the GUI.
- It does not supply any other extendibility method. The only way of extending the tool is to modify the source code
- It uses the file input/output mechanism by using a database. The tool also accepts some standard files as data input and writes this input to some of the tables of the database.

The first level DFDs of the database administration applet and course registration applet are shown in Figure 4.1 and Figure 4.2. In both of the applets first the user-id and password are entered by the user and verified by the tool from the related table or tables in the database and then the user is allowed to use the tool or not according to the result of verification.



**Figure 4.1: First level DFD of database administration applet.**



**Figure 4.2: First level DFD of course registration applet.**

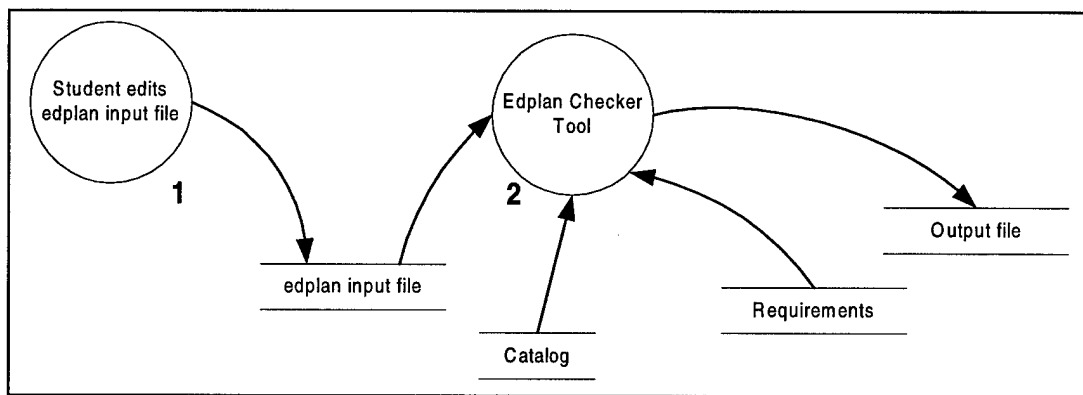
The Edplan administration tool uses the JDBC API for database access. The tool uses the thick JDBC driver supplied by Oracle, which is a type-2 driver. Since this driver uses local native libraries to communicate directly to the database it cannot be used within an applet.

The other tool to be integrated is the Edplan checker tool. The purpose of this tool is to check whether the Edplans of the students comply with the requirements or not. This tool is written in the Prolog programming language and the source code is available. This tool is run in a Prolog compiler named SWI-Prolog [10]. Since the Edplan checker tool is run in the SWI-Prolog environment we have to analyze both the tool and SWI-Prolog compiler.

The SWI-Prolog is a freely distributed, publicly available Prolog compiler. It has versions that run on all of the major platforms including Unix, Windows, and Linux. SWI-Prolog is normally operated as an interactive application simply by starting the program. Then SWI-Prolog enters interactive mode. Here the user can load the Prolog files and run them. SWI-Prolog also has command line parameters to run Prolog programs

without entering the interactive mode. Normally the Edplan checker tool is run in the interactive mode.

Running the Edplan checker program is not an easy process. The student first has to run the SWI-Prolog interpreter. Then he/she has to load the starter program, which then loads the main part of the program. The tool runs in an interactive mode after this point. The student has to have the Edplan information ready as a specially formatted text file. Then the student loads this input file. If there isn't any typing error in the input file the file loads correctly. Then the student gives the command to check the Edplan. If the Edplan of the student meets the requirements in the Edplan checker tool's database the student can save the output of the Edplan checker. The first level DFD of the Edplan checker tool can be seen in Figure 4.3.



**Figure 4.3: First level DFD of Edplan Checker tool.**

The source code of the Edplan checker tool is available. So, it can be extended by modifying the code or adding new code to it. Then we can say that the Edplan checker tool is extendable through source code availability. There are two parts of the source code. The first part loads the main part of the program. This first part is just used as a starter program. The tool does not supply a way of extendibility other than modifying the

code. Now we have to do an analysis of the source code of the program. The starter program file is "edplan95.pl". This part of the program just loads the main program file "main.pl" and gives information to the user to type "begin" and start the program. After loading the files and the database and giving some initial information to the user the tool enters to the interactive mode. In the interactive mode the tool waits for commands from the user and runs according to the commands given by the user.

The tool uses files for input and output as explained above. Then the input/output mechanism of the tool can be characterized as "file input/output." The program loads the student's Edplan data as a formatted text file. The catalog data about the courses given and the requirements data about these courses are loaded from text files. After the student loads and checks the Edplan he/she can save the output of the Edplan checker as an output file. This output file has the same name as the input file. But the program gives an extension of ".out" to the output file. The Unix version of the SWI-Prolog compiler uses standard input and standard output when it is in interactive mode. So, when SWI-Prolog is run on a Unix platform it also uses the standard I/O. Then we can also characterize the I/O mechanism of the tool as "standard input/output."

As the last step of this initial tool analysis we have to characterize the tools as a pair. When we consider the tools as a pair we are going to characterize the tools as "both extendable". The results of the tool analysis can be seen in Table 4.1.

Tool	Extendibility		Input/Output	User interface
Edplan Administration Tool	Source code	Both extendable	File I/O GUI	GUI
Edplan Checker Tool	Source code		File I/O Standard I/O	Interactive interface

Table 4.1: Tool Characteristics Summary

## 4.2. Choice Of Integration Method

While deciding on the integration method that we are going to use for this integration effort we are going to use the results of the tool analysis as defined in the methodology. As we can see in Table 4.1, both of the tools are extendable through source code and they don't supply any other extendibility method. In such a case we can choose any of the three paradigms. Our choice of integration method must be the one that is easiest to implement and the one that best fits our system. One of the tools that we are trying to integrate is a Java applet and the other is a Prolog program. The Edplan administration tool is written in the Java programming language, which is the programming language of the Internet. This tool has lots of flexibility. It can run on any operating system or any hardware platform. Because the tool is designed as a Java applet it can be run on the client side within the browser. But some components of the tool can be run on the server side and the tool can communicate with them over the Internet using the HTTP protocol. Since the source code is available we can extend the tool for adding new functionality to ease the integration process. Because this tool is more compatible with the Internet platform than the Edplan checker tool, we will choose the first paradigm for the integrated system. We will extend the Edplan administration tool to include the integration func-



tions and control the Edplan checker tool from the Edplan administration tool. When we use the first paradigm the integrated system will be as in Figure 4.4.

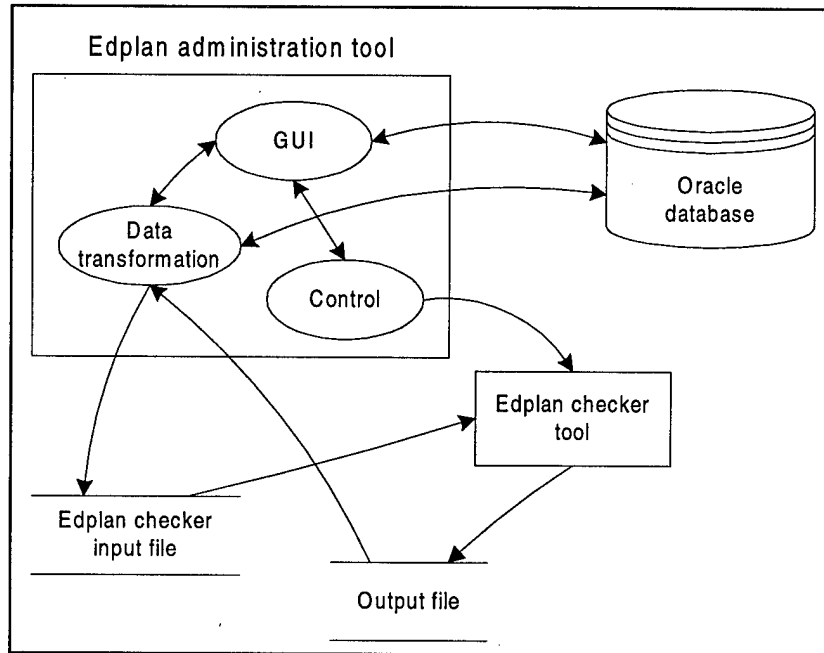


Figure 4.4: System design using the first paradigm.

### 4.3. Analysis Of The Target Platform

We are going to list and analyze the available web servers, firewalls and proxy servers, and the application servers in this part. There are two main points of view for the target platform; the servers available and visible inside of the organization and the servers that are visible outside of the organization. We have lots of different options for serving inside of the organization. We will consider three different web servers for the purpose of this integration effort. These are Sun's Java Web Server Version 2.0 [11], Apache Group's Tomcat, which is part of Jakarta project [12], and Apache Group's Apache Web Server [13]. These three web servers have different properties. We will list the properties of each of them.

Sun's JavaWebServer is a crossplatform web server, which can be installed and run on any JDK 1.1 or Java™ 2 compatible platform. It is a pure Java web server. It is based on the following standards; Java™ 2 platform, Java Servlet API 2.1, Java Server Pages 1.0, HyperText Transfer Protocol (HTTP) 1.1, Secure Socket Layer (SSL) 2.0, and Common Gateway Interface (CGI) 1.1. Even though it has support for all platforms that support JDK 1.1.7, platforms other than Solaris and Windows are untested and therefore unsupported. It has an applet-based, easy to use administration utility. The web server can be monitored or configured from any JDK 1.1 compliant web browser by using this administration utility. This administration applet simplifies the administration tasks. There is a 30-day trial version of JavaWebServer 2.0 that can be downloaded from Sun's web site [14]. Since it is easy to install and easy to administer it can be used for test purposes. But a licensed version has to be bought in order to use this product for deployment of the system.

Tomcat is another web server, which is a combined JSP 1.1 and Servlet 2.2 reference implementation, being developed under the Apache web server . Tomcat can be run together with any web server. It gives the ability to process JSP and servlets to the web server. Because it has a lightweight web server within itself, it can also be used as a standalone web server without using any other web servers [12]. Tomcat is very useful for developing JSP and servlets. Because it can be run within the VisualAge for Java development environment we can run and debug servlet and JSP code [15]. This is a very important property for this development environment, because there aren't many development environments that let the developer debug servlet and JSP code. Tomcat is a very good environment for the development, debug, test and presentation of a developed sys-

tem. But Tomcat alone is not appropriate for deployment. It can be used together with the Apache web server, which is one of the most widely used web servers. Apache web server has support for many platforms including almost all major Unix platforms, Linux, and Windows NT.

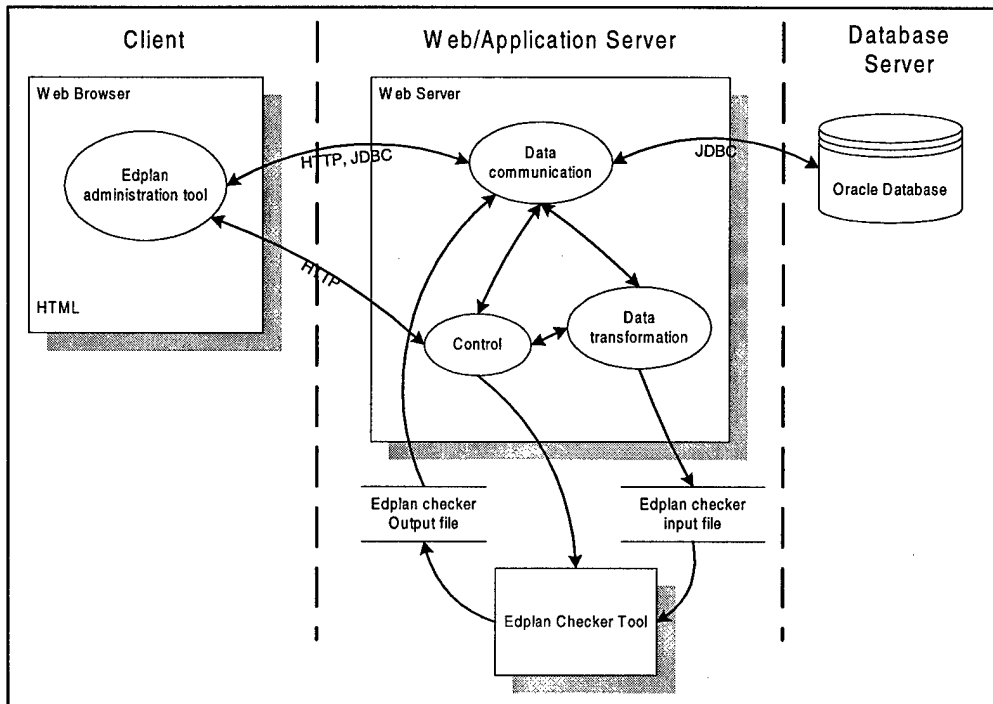
The main and official web server of AFIT, which is visible outside of AFIT to the Internet, is a Microsoft IIS. This web server has support for Microsoft technologies like ASP, ADO, and ActiveX components. But it does not have support for server side java technologies like JSP and servlets, at least for now.

For the purpose of this project there isn't any need for an application server, and there isn't one available. AFIT does not have a proxy server. But there is a proxy server and firewall for Wright Patterson AFB, which is out of AFIT. Since this project is not intended to be served outside AFIT, there is no need to worry about this proxy server and the firewall.

Because it is very easy to maintain and it allows debugging and tracing of the servlet and JSP code during development time, we will use the Tomcat web server for development and presentation of this project.

#### ***4.4. System Design***

In the second step of the methodology we have chosen the first paradigm for the design of our system. Since the Edplan administration tool is a Java applet we can run this tool on the client side. We can run the Edplan checker tool on the server side. This can be the same computer with the web server or it can be another computer on the network. The design of the system can be seen in Figure 4.5. This system design is like the one shown in Figure 3.6.



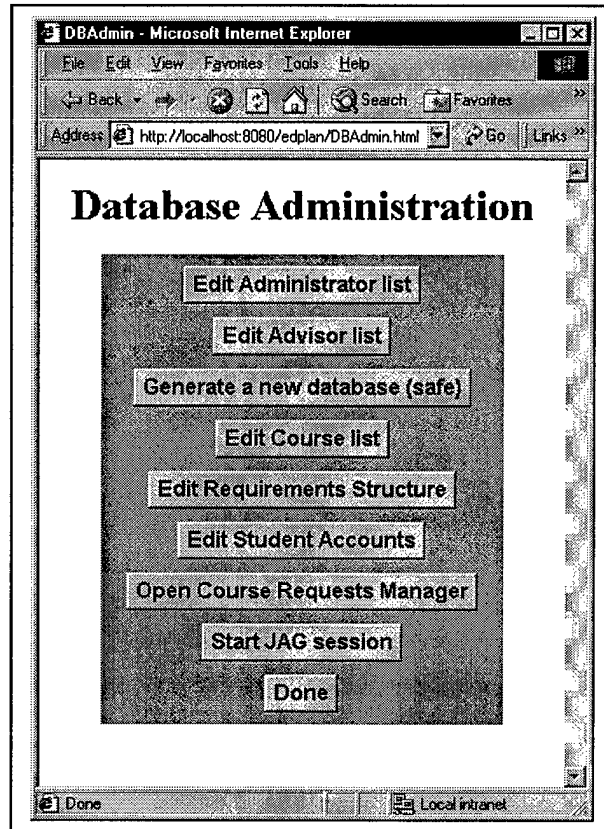
**Figure 4.5: Design of the integrated system using the first paradigm and the first tool on the client side.**

According to the methodology defined in Chapter 3 we have to make a detailed analysis of the source code of the tool here. This analysis will include the following steps:

- Analysis of the data flow within the tool
- Analysis of the input and output mechanisms
- Analysis of the database
- Analysis of the source code according to the security restrictions

*Analysis of the data flow:* The tool has two main parts, each designed as a separate applet. One of them is for the database administrator to maintain the database (DBAdmin.java), and the other part is for the students to register for the courses (register.java). Both of these parts are AWT based Java applets. We are going to make the

analysis of these parts separately. We will start with the analysis of the database administrator part. The user



**Figure 4.6: Main database administration page**

first has to logon to the system by entering his/her user id and password through the logon window. This applet shows different menu buttons according to the person who uses it. If the user is a database administrator then the applet will display a menu as shown in Figure 4.6. If the user is not a database administrator, but an advisor then a restricted version of this menu will be displayed.

After the validation of the user id and password by accessing to the database the main administration menu is displayed as shown in Figure 4.6. By using this main administration menu the user can get into the several different modules of the tool. These

modules are: Administrator list editing, Advisor list editing, Requirement structure editing, Course list editing, Student account editing, Request manager, new database generation, and Edplan administration modules. Each of the modules runs in a separate window. Each of the buttons on the main database administration applet opens a new window frame. Within each module the tool accesses the database many times and the data is passed back and forth between the database server and the client.

The registration applet is for the students. The students use this applet to register for the courses. The first time a student uses this tool he/she enters data about himself/herself by choosing the “New student” button from the Logon window. Then the main menu is displayed as shown in Figure 4.7. The student can modify his/her account information and register for courses by using this menu.

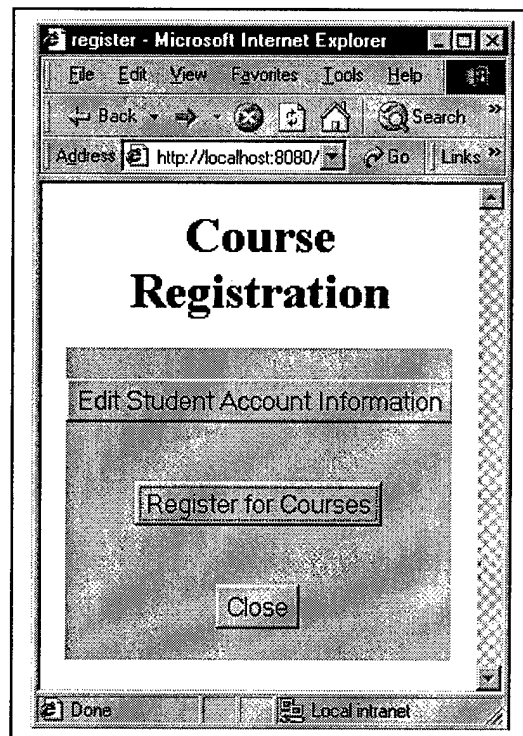


Figure 4.7: Course registration applet window

*Analysis of the input/output mechanisms:* The tool uses a file input/output mechanism considering the database access as a file input/output. The user data input to the tool is through the GUI. The tool also uses some files for the data input to some of the tables in order to make the data input faster. The tool also uses the GUI as the output mechanism. So we are going to characterize the tool's input/output as file and GUI input/output mechanism.

*Analysis of the database access:* As explained above, the tool uses the Oracle DBMS for data storage. The tool uses the JDBC API for accessing the database. It uses the thin JDBC driver provided by Oracle. This driver is a Type 4 driver [16]. It makes direct socket connections to the database server in order to access the database. The tool lets the database administrator create twelve initial tables on the first run of the tool by pressing the "generate database" button. The schema definitions of these tables are in Appendix A. The tool creates several different tables according to the requirement structure created within the tool. These tables keep the data about the courses and the requirement structure about the courses.

*Analysis of the source code according to the security restrictions:* The tool is written using Java applets using the AWT of the Java programming language as explained above. An applet is generally created to run within a browser window. To be able to run an applet within the browser the applet must conform to some security rules and restrictions. These rules are explained in Chapter 2. The Edplan administration tool breaks these rules at several places within the code. The following security problems within the tool prevent it from running within a browser. [6]

1. The tool uses a Type 4 JDBC driver to access Oracle DBMS. This JDBC driver makes socket connections to the server to make the communication possible between the client and the server. But the security restrictions prevent an applet from making a socket connection to a server other than the one it is loaded from. In order to overcome this restriction we either have to change the JDBC driver used to access the database or run the web server on the same computer with the database server.
2. There are several places within the source code of the tool that the `System.exit()` command is used. This command closes the java virtual machine (JVM) that runs the application. The use of this command is restricted within an applet because this command tries to shut down the JVM of the browser when used. This is not allowed for an applet that runs within a browser and causes a security exception.
3. There some portions of the source code, in which the tool tries to access the local file system to load some files. Applets are not allowed to read from or write to the local file system. These portions of the code will also cause security exceptions.

In order to solve the database connection problem we have to use a type-3 JDBC driver, which will use some type of middle tier to communicate with the database server. We use a type-3 JDBC driver distributed with a book about Java servlets [17]. This driver uses Java servlets as the middle tier. This JDBC API has two parts. One part is on the client side and the other part is on the server side. The part on the client side is a standard JDBC API. But this part sends the database requests to the part on the web server instead



of sending them directly to the database server. While sending the JDBC requests to the web server this driver wraps the requests in the HTTP protocol. There are Java servlets on the web server that get the JDBC requests in the form of HTTP requests from the client, change them to JDBC calls to the database and send them to the database server. While doing this these servlets use another JDBC driver. This JDBC driver can be any driver that can be specified by the system designer. In our case the servlets use the thin JDBC driver supplied by Oracle that was originally used by the Edplan administrator tool. These servlets get the results from the database and pass them to the client using the HTTP protocol.

To get rid of the second security problem we have to remove the `System.exit()` commands from the code. We have to change these lines so that the tool opens a message box saying that something wrong has happened and then either continue to run the tool or load an HTML page from the web server.

In order to overcome the third security problem we either need to remove these lines of code or change the tool so that it accesses the server resources to load or save the file. To do this we have to add some server side components that will do this on the web server. These components will be some Java servlets. These servlets will get the file request from the client, read the file from the server file system, and then send the file to the client using the HTTP protocol.

We will need to design the data communication between the tools. As we see in Table 4.1 both of the tools use File I/O mechanism. According to Noe we can use a shared data communication system in this case. Edplan Administration Tool writes its output to a file and we run the Edplan Checker tool so that it loads this file and writes its

output to another file, which can be read by the first tool. The input file has to be in proper format for the second tool. Since both of the tools are extendable we can add the data transformation functions in any of the tools. Based on familiarity with the Java programming language the Edplan Administration tool is extended to make the data transformation.

Another thing that we have to design is the component that will control the Edplan checker tool. As we have found out in the first level analysis of the tools the Edplan checker tool has an interactive interface and it uses file I/O and the Unix and DOS versions use both standard I/O and file I/O. We have two options here for the control mechanism of the Edplan checker tool. The SWI-Prolog compiler can run the Prolog programs without entering the interactive mode by giving a top-level goal with command line parameters. If we can modify the source code of the Edplan checker tool so that it can be run in this way we can add a server side component to the Edplan administration tool, which will run the Edplan checker tool with a system command and with necessary command line parameters. The other option for the control of the Edplan checker tool is to use the standard input and standard output to control the Edplan checker tool.

First I tried to modify the Edplan checker tool's code in order to be able to run it without entering the interactive mode. But I couldn't successfully do this. I knew that this could be done. But, because of the unfamiliarity with the Prolog programming language I couldn't modify the code successfully. The second option was to use the standard input/output to run the Edplan checker tool. I added a server side component to the Edplan administrator tool. This server side component is a Java servlet that runs the Edplan checker, sends the necessary commands to the tool to load the input file and save the out-

put file using the standard input, and pipes the output of the SWI-Prolog compiler from the standard output to the client. This way the user can interact with the Edplan checker tool remotely and run it with the Edplan checker input file created by the Edplan administration tool.

#### ***4.5. Summary***

In this Chapter application of the methodology defined in previous Chapter is demonstrated through integration of the Edplan creation and administration tool (CLAS-PICS) and the Edplan checker tool. The two tools are integrated to run in a web-based environment. The Edplan creation and administration tool can be downloaded from the web server and run within a web browser. The Edplan checker tool can be accessed and used from the Edplan creation and administration tool in the new system.

## **5. Results and Conclusion**

This research briefly examines web-based applications and the trend on development of web-based applications, “web-enabling” existing applications and the integration of tools and applications in web-based environments. Integrating old tools and applications into the web-based environment instead of re-writing them is an easier and less costly way. The primary objective of this research was to develop a methodology for integrating tools in a web-based environment. This objective was accomplished through the development of a generic methodology for integrating existing tools in a web-based environment.

After doing a literature review and research on web technologies currently used in developing web-based applications, it was determined that the technology in this realm is developing very rapidly and new technologies are being introduced very frequently. There is an increasing trend towards Java technology, which has been proven to be the programming language of the Internet and web-based applications. Java based technologies are also improving very rapidly. The rest of this chapter summarizes the work accomplished during this research effort. The results of the sample application of the methodology are discussed and some recommendations for future work are given.

### **5.1. Results**

The web-based tool integration methodology developed in this thesis research offers a step-by-step approach to web-based tool integration. The two tools that are going to be integrated share some information. The information that the tools use may not be in the same format. But it must be semantically meaningful to both tools. The methodology assists the integrator in analyzing the tools separately and as a pair. The results of the

analysis will help the integrator during the integration process and in designing the resulting integrated system. According to the results of this first level analysis the tools will be characterized with respect to their input/output mechanisms and extendibility. Using the results of this first level analysis and tool characterization an integration method is chosen. There are three integration paradigms outlined in Chapter 3. Choice of an integration paradigm is done according to the extendibility class of the tool pair. The data communication and data transformation needs between the tools are also clarified while choosing the integration method. Then we analyze the target platform according to the available web servers, firewalls and proxy servers, and application servers. Then the system design is done according to the integration method we have chosen.

This methodology is used to integrate two tools that are used in the course registration process at AFIT. These tools are the Edplan creation and administration tool developed by Jason Gunsch and the Edplan checker tool developed by Dr. Gunsch. The Edplan creation and administration tool is a tool that helps students while creating their Edplans and saves the Edplans in a relational database. The Edplan Checker tool is used by the students to check their Edplans to determine if they comply with the requirements or not. The purpose of the integration is to integrate the tools in a web-based environment in order to web-enable the tools. As part of the integration effort the Edplan creation and administration tool was modified and extended. The tool was modified to make it possible to run it within a browser without causing any security exceptions and to be able to communicate with the database server without any restrictions. It is also extended to make it possible to control the Edplan checker tool from the Edplan creation and administration tool. This way the students can run the Edplan creation and administration tool

and the Edplan checker tool at the same time without switching the environments or without any need to learn the input file syntax for the Edplan checker tool. The Edplan administration tool can now be run within a browser by downloading it from a web server and it can communicate with the database server without any problems. The integrated system is designed so that the Edplan creation and administration tool has some parts on the web server that will control the Edplan checker tool and make any data transformation between the tools. The control integration component of the Edplan creation and administration tool can now save the input file for the Edplan Checker tool to the file system of the server, run the Edplan checker tool with the saved input file, read the output of the Edplan Checker tool, and send them to the client to be displayed to the user.

## ***5.2. Conclusions***

The primary objective of this thesis research was to develop a methodology that would provide a step-by-step methodology for integrating tools in a web-based environment. The developed methodology has demonstrated to be appropriate and good enough for integrating tools in a web-based environment. The two tools are integrated effectively in a web-based environment to form a web-based system. The system design is developed with the help of the methodology. The methodology assists the integrator during the design of the integrated system. But it does not deal with the implementation of the designed system. Implementation is left to the integrator. The integrator has to find an appropriate implementation method according his/her knowledge and experience.

## ***5.3. Recommendations For Future Work***

The methodology developed during this thesis research accomplished quite a bit for integration of tools in a web-based environment. It also helped in the design of an in-

tegrated system for AFIT Edplan administration. Both the methodology and the designed system can be improved in many ways. The rest of this section will give some recommendations for improvement of the methodology and for the AFIT Edplan administration and course registration system.

### **5.3.1. Recommendations For The Methodology**

The methodology generated can always be extended to reach perfection. It will need much work to be perfect. As mentioned many times in previous chapters, the technology in the realm of web-based applications and the Internet is improving rapidly. New technologies are introduced frequently. These new technologies can be used to improve the methodology developed here. One of these new technologies is XML (Extensible Markup Language). XML is a new technology for web applications. It is a W3C (World Wide Web Consortium) standard that lets a developer build his own tags. XML is a breakthrough for the data interchange between different organizations or even different parts of the same organization. XML makes it easy to send structured data across the web so that nothing gets lost in transaction [18].

The methodology developed in this research does not address integration of the databases that the tools use. Since the tools use related information, they probably have the same information in different formats and different database systems. The methodology can be extended to address the integration of the databases that the tools use so that the tools start using the same database. This way information can be centralized, more consistent, and easy to maintain.

### **5.3.2. Recommendations For The Course Registration System**

The course registration system in AFIT can be improved and extended in many different ways. One of these is integrating the Edplan administration tool with a course-scheduling tool. Course scheduling is one of the problems in AFIT as in many different educational organizations. Currently there is some research on developing a course scheduling system for AFIT. There are also some tools developed previously that do course scheduling. Integrating these tools with the current system could be a very important improvement for the system. Another improvement for the system would be the integration or synchronization of the current system with the STARS database.

There are some APIs that are used to integrate the Prolog programming language to other programming languages and environments. These APIs allow the use of the Prolog programming language within other programming languages. One of these is JPL [32]. JPL is an API that uses JNI (Java Native Interface) to integrate Prolog and Java. JPL supports the embedding of a Prolog engine within the Java VM. Another one is AMZI! Prolog [20]. This one also supplies an API to access Prolog from Java.

An API like this could be used to seamlessly integrate the Edplan checker tool to the Edplan administration and course registration tool. The Edplan checker tool and the Edplan administration tool use different databases to store information about the courses. These databases could also be integrated into one database.



## Appendix A. Table Definitions

### ADMINISTRATORS

Field Name	Null?	Type
ADMINID	NOT NULL	VARCHAR2 (15)
PASSWORD		VARCHAR2 (32)

### ADVISORS

Field Name	Null?	Type
ADVISORID	NOT NULL	VARCHAR2 (15)
PASSWORD		VARCHAR2 (32)
NAME_LAST	NOT NULL	VARCHAR2 (40)
NAME_FIRST	NOT NULL	VARCHAR2 (20)
MIDDLENAME		VARCHAR2 (20)
SSN		CHAR (11)
DEPARTMENT	NOT NULL	VARCHAR2 (32)
GRADE		VARCHAR2 (15)

### COLLEGEREQUIREMENTSTABLES

Field Name	Null?	Type
CREQ	NOT NULL	VARCHAR2 (32)

### COLLEGES

Field Name	Null?	Type
COLLEGE	NOT NULL	VARCHAR2 (32)
CREQTABLE		VARCHAR2 (32)

### COURSES

Field Name	Null?	Type
PREFIX	NOT NULL	CHAR (4)
CODENUM	NOT NULL	CHAR (3)
TITLE	NOT NULL	VARCHAR2 (100)
DEPT		CHAR (3)
CDESCR	NOT NULL	VARCHAR2 (10)
CREDITMIN	NOT NULL	NUMBER (38)
CREDITMAX	NOT NULL	NUMBER (38)
PREREQ	NOT NULL	VARCHAR2 (100)
COREQ	NOT NULL	VARCHAR2 (100)
DESCR	NOT NULL	VARCHAR2 (2000)
QUARTERS	NOT NULL	VARCHAR2 (15)
REPEATABLE	NOT NULL	CHAR (1)

### OFFICIALCOURSELIST

Field Name	Null?	Type
NETID	NOT NULL	VARCHAR2 (15)
PREFIX	NOT NULL	CHAR (4)
CODENUM	NOT NULL	CHAR (3)
CREDIT		NUMBER (38)
STATUS		CHAR (2)
REQUIREMENT		VARCHAR2 (32)
SEQUENCE		VARCHAR2 (32)
QUARTER		CHAR (4)
VARIABLECREDIT		CHAR (1)

### PERMITTEDSEQUENCESTABLES

Field Name	Null?	Type
PERSEQTABLE	NOT NULL	VARCHAR2 (32)

### REQUESTEDCOURSELIST

Field Name	Null?	Type
------------	-------	------

NETID	NOT NULL VARCHAR2 (15)
PREFIX	NOT NULL CHAR (4)
CODENUM	NOT NULL CHAR (3)
CREDIT	NUMBER (38)
STATUS	CHAR (2)
REQUIREMENT	VARCHAR2 (32)
SEQUENCE	VARCHAR2 (32)
QUARTER	CHAR (4)
VARIABLECREDIT	CHAR (1)

REQUIREMENTSTABLE

Field Name	Null?	Type
REQUIREMENT	NOT NULL	VARCHAR2 (32)

SEQUENCESTABLE

Field Name	Null?	Type
SEQUENCE	NOT NULL	VARCHAR2 (32)

SPECIALINFO

Field Name	Null?	Type
TABLENAME INFORMATION	NOT NULL	VARCHAR2 (32) VARCHAR2 (2000)

STUDENTS

Field Name	Null?	Type
NETID	NOT NULL	VARCHAR2 (15)
PASSWORD	NOT NULL	VARCHAR2 (32)
NAME_LAST	NOT NULL	VARCHAR2 (40)
NAME_FIRST	NOT NULL	VARCHAR2 (20)
MIDDLENAME		VARCHAR2 (20)
SSN	NOT NULL	CHAR (11)
INYEAR	NOT NULL	NUMBER (38)
DEPARTMENT	NOT NULL	VARCHAR2 (32)
ACADEMICSPECIALTYCODE	NOT NULL	CHAR (4)
ADVISORID	NOT NULL	VARCHAR2 (15)
NEWREQUEST		CHAR (1)
GRADE		VARCHAR2 (15)
QUOTA		VARCHAR2 (8)
THESISQUARTER		CHAR (4)

## Bibliography

- [1] Lee, Tien-Chen  
*A Web-based prototype for AFIT Edplan Administration*, Thesis,  
Air Force Institute of Technology, Wright-Patterson AFB, OH, Dec 1998  
AFIT/GCS/ENG/98D-02
- [2] DiPiro, Daniel L.  
*Methodology for the Analysis and Design of Internet Software Component Providing Relational Database Access Through the World Wide Web*, Thesis  
Air Force Institute of Technology, Wright-Patterson AFB, OH, Dec 1998  
AFIT/GCS/ENG/98M-01, AD-A340958 DTIC
- [3] Noe, Penelope A.  
*A Structured Approach to Software Tool Integration*, Thesis  
Air Force Institute of Technology, Wright-Patterson AFB, OH, Dec 1999  
AFIT/GCS/ENG/98M-14
- [4] <http://jserv.javasoft.com/products/java-server/servlets/environments.html>  
JavaSoft Web Site
- [5] [www.chilisoft.com](http://www.chilisoft.com)  
ChiliSoft Web Site
- [6] Weber, Joseph L.  
*Using Java 1.2*, p. 777, QUE Publishing, 1998
- [7] Weiss, Aaron  
*Introduction to DHTML*, 24 August 1998,  
<http://www.wdvl.com/Authoring/DHTML/Intro/>
- [8] Wasserman, Anthony I.  
*Tool Integration in Software Engineering Environments*,  
Software Engineering Environment: Proc. International Workshop on Environments, F.Long, ed., Springer-Verlag, Berlin, 1990
- [9] Thomas, Ian and Brian A. Nejme.  
*Definitions of Tool Integration for Environments*,  
IEEE Software, pp. 29-35, March 1992
- [10] <http://www.swi.psy.uva.nl/projects/SWI-Prolog/>  
SWI-Prolog Web Site

- [11] Java Web Server Version 2.0  
Sun Microsystems Web Site  
<http://www.sun.com/software/jwebserver/index.html>
- [12] Tomcat Web Server, Jakarta Project,  
Apache Software Foundation  
<http://jakarta.apache.org/tomcat/index.html>
- [13] Apache Web Server  
Apache Software Foundation  
<http://www.apache.org/httpd.html>
- [14] Sun Microsystems Web Site  
<http://www.sun.com>
- [15] Wosnick, Sheldon  
*Apache Tomcat Servlet and JavaServer Pages Development with VisualAge for Java,*  
<http://www7.software.ibm.com/vad.nsf/Data/Document2389?OpenDocument&p=1&BCT=3&Footer=1>
- [16] Oracle 8i Online Documentation
- [17] Moss, Karl  
*Java Servlets Second Edition*, McGraw-Hill, 1999
- [18] Tidwell, Dough  
*Tutorial: Introduction to XML,*  
XML developerWorks Team, Raleigh, NC, July 1999,  
<http://www.ibm.com/developer/>
- [19] Bosak, Jon  
*XML, Java, and the future of the Web*, Sun Microsystems, March 1997,  
<http://metalab.unc.edu/pub/sun-info/standards/xml/why/xmlapps.htm>
- [20] Amzi! Prolog,  
Amzi! Inc. Web Site,  
<http://www.amzi.com>
- [21] Microsoft Developer Network Library,  
*ODBC Programmer's Reference*
- [22] Graham, Ian S.  
*HTML Sourcebook*, John Wiley & Sons, Inc. 1995

- [23] Dobson, Rick  
*From Access to ASP to Web: Understanding Active Server Pages: Part I*  
<http://msdn.microsoft.com/library/periodic/period98/html/OVBAD0298asp.htm>
- [24] Garris, John  
*Scripting with ASP*, PC Magazine
- [25] Lam, John  
*Active Server Pages*, PC Magazine, September 21, 1998
- [26] Karpinski, Richard  
*Dynamic Java Pages, MS-Style*, Internetweek, June 29, 1998, Issue 721
- [27] Sol, Selena  
*Web Programming 101 Part Two: Client Side Scripting*  
WEB Developer's Virtual Library  
<http://www.wdvl.com/Authoring/Scripting/WebWare/Client/index.html>
- [28] *Client Side JavaScript Guide*, Chapter 1, JavaScript Overview  
Netscape Communications Corporation 1999  
<http://developer.netscape.com/docs/manuals/js/client/jsguide/intro.htm>
- [29] Mohseni, Piroz  
*Start serving Java Server Pages: Introduction*, Gamelan, Tech Workshop  
[http://www.gamelan.com/journal/techworkshop/092199\\_jsp1.html](http://www.gamelan.com/journal/techworkshop/092199_jsp1.html)
- [30] Hamilton, Graham; Cattell, Rick; Fisher, Maydene  
*JDBC Database Access with Java*, Addison Wesley, January 1998
- [31] *Building Java Applications for the Oracle Internet Platform With Oracle JDeveloper*, An Oracle White Paper, October 1999  
[http://technet.oracle.com/products/jdev/htdocs/jds\\_oip.htm](http://technet.oracle.com/products/jdev/htdocs/jds_oip.htm)
- [32] Dushin, Fred  
*JPL: A Java interface to Prolog*  
<http://blackcat.cat.syr.edu/~fadushin/software/jpl/>

**REPORT DOCUMENTATION PAGE**

*Form Approved*  
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

**PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

<b>1. REPORT DATE (DD-MM-YYYY)</b> 21April 2000	<b>2. REPORT TYPE</b> Master's Thesis	<b>3. DATES COVERED (From - To)</b> APR 1999 - APR 2000
--	--	--

<b>4. TITLE AND SUBTITLE</b>  A METHODOLOGY FOR INTEGRATING TOOLS IN A WEB-BASED ENVIRONMENT	<b>5a. CONTRACT NUMBER</b>
	<b>5b. GRANT NUMBER</b>
	<b>5c. PROGRAM ELEMENT NUMBER</b>

<b>6. AUTHOR(S)</b>  Arslan, Musa Serdar, 1st Lt., TUAF	<b>5d. PROJECT NUMBER</b>
	<b>5e. TASK NUMBER</b>
	<b>5f. WORK UNIT NUMBER</b>

<b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</b> Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 P Street, Building 640 WPAFB, OH 45433 - 7303 DSN: 785-2811 X 4364	<b>8. PERFORMING ORGANIZATION REPORT NUMBER</b>  AFIT/GCE/ENG/00J-01
--	--

<b>9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b> AFIT/RAD Attn: Baker, Randall B. Air Force Institute of Technology, Registrar Division Ph: 937/255-3094	<b>10. SPONSOR/MONITOR'S ACRONYM(S)</b>
	<b>11. SPONSOR/MONITOR'S REPORT NUMBER(S)</b>

**12. DISTRIBUTION/AVAILABILITY STATEMENT**  
  
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

**13. SUPPLEMENTARY NOTES**  
  
Professor Thomas C. Hatrum, CIV AFIT/ENG, Ph: 937/255-3636 x 4581

**14. ABSTRACT**  
Web-based technologies are evolving very rapidly. New technologies are introduced very frequently in the realm of the Web and Internet. This evolution also affects database management systems (DBMSs). Almost all DBMS vendors are making their systems "Web-enabled". The Internet and the World Wide Web are getting more important and bigger than ever. Because of the increase in the importance of the Internet and the Web, migrating old applications and tools to a web-based environment is becoming more important. When migrating old applications or tools to a web-based environment integration of tools becomes an important issue. In this research, a step-by-step methodology for integrating tools in a web-based environment is developed while trying to integrate two tools that are part of the AFIT education plan administration and course registration system. These two tools are CLASPICS (Computerized, Lightweight Assistant for Student Programme Identification and Course Selection) tool and Education plan checker tool. The methodology proved to be effective for integrating tools in a web-based environment.

**15. SUBJECT TERMS**  
Web-based tool integration methodology, tool integration methodology, web-based environment

<b>16. SECURITY CLASSIFICATION OF:</b>			<b>17. LIMITATION OF ABSTRACT</b>  UL	<b>18. NUMBER OF PAGES</b>  94	<b>19a. NAME OF RESPONSIBLE PERSON</b> Thomas C. Hartrum, Professor, CIV AFIT/ENG
<b>a. REPORT</b> Unclassified	<b>b. ABSTRACT</b> Unclassified	<b>c. THIS PAGE</b> Unclassified			<b>19b. TELEPHONE NUMBER (Include area code)</b> 937/255-3636 x 4581