CHAPTER I

INTRODUCTION

Computer Simulation

Computer simulation is the discipline of designing a model of an actual or theoretical physical system, executing the model on a computer, and analysing the execution output [Fishwick, 1995]. In general, simulation is to pretend that one deals with a real thing while really working with an imitation. In operational research the imitation is a computer model of the simulated reality. Models are used in industry and commerce and military because to make experiment with real systems is very costly, dangerous and sometimes impossible. Provided that models are adequate descriptions of reality, experimenting with them can save money, suffering and time [Pollatschek, 1999].

There are generally two types of simulation:

- **Discrete Event Simulation**. When the numbers of events are finite and between two consecutive items nothing happens, we call this kind of simulation a discrete event. Car arrivals and departures occurred at distinct points of time are example of discrete event simulation.

- **Continuous Simulation**. In some systems the state changes all the time, not just at time of some discrete events. For example water level in a reservoir with given in and outflow may change all the time.

Simulation is often performed by writing a computer code to represent a system model, or as some kind of input into a simulator software. Simulation may be carried out:

- On a spreadsheet
- By running a computer program written in some general language (such as Fortran, C/C++ or Pascal)
- By running a computer program written in some special language for simulations
- By running specially built simulator with appropriate input.

These four methods are the most popular methods of performing simulations. But due to the dependency on the local standalone system they are running on, these simulation methods suffers several disadvantages such as platform-dependent, machine-dependent, and require longer learning curve. With the current advancement in Internet and web technology, we propose a partial solution to the above problems with a Web-based Simulation Environment (Websim) project. Websim provides a dynamic web interface for a standalone simulation engine and allow the performing of simulation works over the World Wide Web (WWW). Websim also provides a simulation environment where users can examine the simulation outputs to verify and validate their system model. The outputs can either be in text or graphical representation.

Web-based Simulation

The Internet is a loosely connected world-wide network of heterogeneous computers and the WWW is a set of on-line hypermedia documents that resides on the Internet. The WWW was conceived as a set of simple Internet-based client/server protocols for transferring and rendering documents of a primarily textual nature. What distinguished the web’s mode of communicating information from other Internet-based tools that preceded it (e-mail and FTP) was the provision for embedding hyperlinks that allows users to easily navigate between related documents. Hypermedia documents are built using a language called Hypertext Markup Language (HTML). Access to the web is afforded using the web browser such as Netscape Navigator or Internet Explorer.

Web-based simulation was introduced to perform all steps needed for simulation over the Internet and the web. In other words, it uses the web as a means of access to simulation models as well as its use as a platform for the creation, execution and distribution of simulation models [Nam and Kim, 1999].

Most of the existing Web-based simulation tools focused mainly on the development of the runtime simulation libraries and mechanisms on the Web, such as Simjava [Howell, 1999], Javasim [Javasim, 1999], Silk [Healy and Kilgore, 1998] and the development of the distributed simulation environments using ORB technology such as JavaRMI and CORBA [Orfali and Harkey, 1997]. Some of these tools require simulation model developers to be good at C/C++, Java and simulation languages. However, performance of web applications developed in Java are relatively slow in speed. To ease the burden of modellers, GUI based modelling environments are essential. Simulation tools like OPNET [MIL3, 1997] and Digital Workshop [Fishwick, 1998] support the visual development environments, but OPNET could not support Web-based simulation capability, and Digital Workshop could be used only for digital circuit design.

Most of the existing commercial and non-commercial simulation tools are typically platform-dependent and are not designed to work in the Internet or the Intranet. We acknowledge that it is easier to write a simulation model using a general-purpose language such as C/C++ because the C/C++ is very popular and easy to learn. The only problem with programs written in C/C++ is that they are typically machine dependent and standalone. In this thesis, Websim project acts as the missing jigsaw which encapsulates the standalone programs in C/C++ with a web interface, and allows it to be accessible through the web.

Characteristics of Web-based Simulation

A web-based simulation environment refers to a web application system, which provides a dynamic web interface for a simulation tool; and allows the performing of simulation works over the WWW. This environment is accessible with a web browser and has the following characteristics:

- Supports a modelling tool which is independent of hardware and software platforms
- Supports a graphical user interface for easy modelling
- Supports execution over the web
- Able to store and display the simulation results over the web.

In this thesis we develop the Websim, a simulation environment for self-developed simulation tool. Websim shows how one could extend the capability of simulation tools into the web. Websim allows a simulation model to be uploaded to a web server, and automatically generate a web interface for the simulation model. Websim can be accessed on any platform (Unix, PC or Mac) using a standard web browser. With this simulation environment, an easier and faster integration of simulation and visualization techniques and tools into the Internet can be realized.

Significance of Websim

There are three significant aspects of this thesis. First, it is the ability of Websim to produce a web-based simulator on a standalone simulation engine. Websim simply receive the simulation engine in executable formats and provide a web interface to it. Secondly, Websim act as the online store for a collection of simulation models. For example, researchers from the same group could upload their simulation models to Websim, so that their simulation works could be easily referred and executed by other researchers from other parts of the world. Finally Websim could act as a teaching aid / tools in schools and universities. This would be very helpful especially for courses involving modeling and simulation, where lecturers could ask several students who are good in C/C++ programming to develop simulation engines, and upload them to Websim. These web-based simulation engines could assist other peers in having a better understanding of certain topics or concepts.

Objectives

One important property of Websim is that the execution of simulation runs on a remote server. It is designed to provide distributed and platform independent features to multiple concurrent users. For instance, a user could perform the simulation works at home using the simulator located at his office, while another colleague at the office watches the results shown on the computer screen via a web browser. The aim of the thesis is to develop a server-side web application of a web-based simulation environment for self-developed simulation program by utilizing Common Gateway Interface (CGI) and Javascript technologies.

The objectives of this thesis are as follows:

- To enable a standalone simulation program to be executed by multiple concurrent clients over the web.
- To produce web interface for the simulation program based on its input and output data.
- To develop a server side application which could receive the self-developed simulation program (executables C/C++ program file).
- To provide an interface to support execution of the simulation program over the web.
- To develop some useful server-side administrator modules to manage Websim. This include features such as uploading, editing, maintaining, and the like.
- To develop several simulation program as samples to test and evaluate the performance of Websim.

Organisation of Thesis

This thesis is organised into six chapters. Chapter 1 provides a brief introduction to web-based simulation systems and the significance of Websim. Chapter 2 presents the literature review on various web-based simulation topics. The history and the development of web-based simulation are also included. Chapter 3 describes the design methodology employed and the system architecture defined for Websim. As CGI plays an important role in this project, various issues pertaining to this area are highlighted. The system implementation is described in Chapter 4. The performance testing, evaluation of results and general limitations of Websim are presented in Chapter 5. The final chapter discusses the significance and contribution of Websim, its future works and conclusion of this project.

CHAPTER II

LITERATURE REVIEW

Web-based simulation represents the marriage of web technologies and simulation science. The web's ability to service large and diverse audiences allows the simulation community to legitimately provide models and simulations as end products. Recent advances in web technology have made the web a viable mechanism for performing, publishing and distributing simulation [Leong and Guan, 1998]. In this section, we discuss some technical issues related to such technology.

Potential of WWW to Support Simulation

There are many aspects to simulation as there are for any scientific discipline. Here we only present two aspects: 1) Education and Training, and 2) Simulation Programs.

Education and Training

Most modern simulation packages today have graphical front ends (so called Graphical User Interface – GUIs). This allows a student to learn through virtual world exploration. The web will affect this process in several ways. First, a diskette or CD-ROM has a limited amount of local storage but the web provides us with effectively infinite storage. The "storage" is on the Internet and not limited to one's own machine. Therefore simulation packages that include help text and information about the pieces of the models, can be made available on the web so that they do not require local storage. [Fishwick, 1995].

Simulation Programs.

The most exciting part of web-based simulation is in the simulation itself. A web document may contain videos, interactive simulations, images and audio in addition to the usual text that traditional documents contain. A simulation requires a model and a model requires computing hardware to execute the model. Some parts of a model can be executed on another machine and other parts are executed on local machine where the web browser is located.

Web-based Simulation Environment Service

Web-based simulation environment allows different modes of cooperation between a simulation service provider and a simulation customer. Here we present two modes of services commonly used in web-based simulation.

(i) In one mode, the service provider can offer a simulation and animation (S&A) software warehouse (models and tools) on his web server. Customers can work with modified models or models of their own, prepared on their Web client(s). The models can run applets on the client machine or on the service-provider's server with the results then being pushed to the client.

(ii) In an alternative mode, the service provider build a customized model and carry out a complete simulation study. The customer has Web-based access to the model at all steps of the simulation study. He participates actively in the model design, in the debugging and the validation stage and uses Web-based animation tools for evaluation purposes. He is able to execute his own experimental model by changing parameters in an HTML form and starting experiments on the provider's Web server. He has Web-based access to a database containing all results of the experiments.

Architecture of Web-based Simulation System

This section describes three popular approaches for simulation and animation in the Web. Here, we describe how web-based simulation can be achieved by using the various current web technologies. We could generally categorize them into two groups: the remote and local approaches. In remote web-based simulation, the simulation engine resides on the server, the user will perform the simulation by executing the simulation engine remotely. The two popular solutions are by using CGI and Java Data Server. As for local web-based simulation, the simulation engine is in the Java applet’s form, and has to be loaded to user’s web browser to run. This approach utilises the resources found in the local computer system.

Remote Simulation using CGI

In remote Simulation and Animation (Figure 1), the user specifies values of parameters for a simulation model in an HTML form, submits the form to a server and starts the simulation by pushing the start-button on the form. The Common Gateway Interface (CGI) is used to transfer the data to the server. A CGI script starts the simulation by invoking the simulation engine after the data have been received.

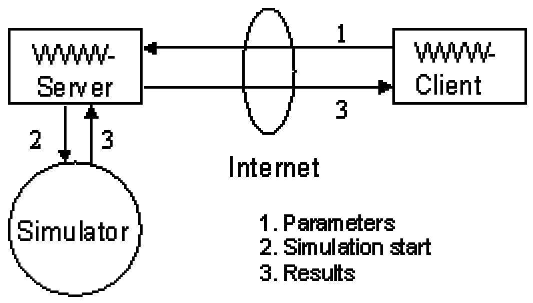


Figure 1: Remote simulation and animation and data transfer

When the simulation finished, the CGI-script prints the results (including the URL of files that have been created to show results) in a new HTML page and is transmitted back to the client. This technique is called dynamic document generation.

The remote approach is suitable for existing S&A software, because it allows the user to create simulation engine, which is independent of web-based simulation environment. Besides, CGI is the most widely used method to perform server-side application, and CGI is fast as it utilizes the resources and power of the server system. However, this approach is not suitable for observing dynamic processes at work and does not allow the user to interrupt a running simulation, since the server sends the results only at completion or at defined points in time.

Remote Simulation using a Java Data Server

This S&A approach, shown in Figure 2, was introduced by Berger and Leiner [Berger and Leiner, 1997]. It uses a WWW and Java server. The simulation runs remotely on a simulation server. The results are transferred to the client and visualized locally.

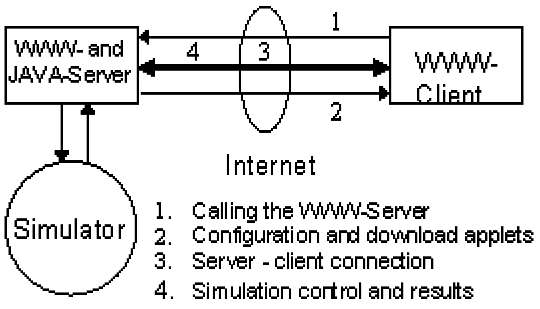

WWW- and JAVA-Server
WWW-Client
1
2
3
4
Internet
Simulator
1. Calling the WWW-Server
2. Configuration and download applets
3. Server - client connection
4. Simulation control and results

Figure 2: Remote Simulation and Local Visualization

The user begins by loading some applets. After these applets have started, a connection to the Java server is built and simulation data are transmitted to the Web browser. The data can change continuously, delayed only by the executing simulation model and transmission time on the Internet. The user can interact with the model by using buttons on the HTML page or by clicking into a graphical representation of the model.

This approach could provide a better performance compared to the CGI approach especially when the number of user is high. This is due to the excellent thread handling features provided by Java language. Java also provide graphical features which can be used to simulate the observing of online dynamic processes. However, the drawback of this approach is it will require a Java web server to be installed and run on the server.

Local Simulation Based on Java Applets

In local S&A based on loading applets (Figure 3), the user loads a Java simulation applet into the Web-browser, which then runs the simulation on the client machine. This approach supports the incorporation of user interaction and animation into the simulation applet. It is suitable for small simulation engine which does not require much heavy processing.

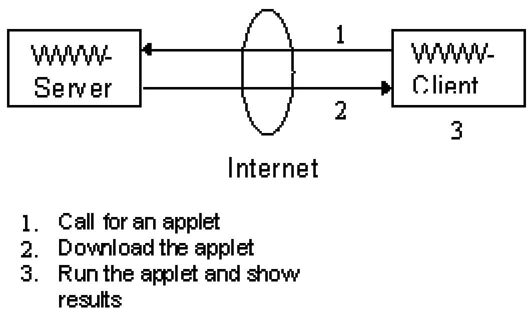


Figure 3: Client-Site simulation with loaded applets

Existing simulation engines are not suitable for this approach. Applets has no direct server accessing features, and therefore would require the simulation engine to be written in Java language. Besides, Applets speed is very slow and dependent on the traffic of the web. Since time is crucial in simulation works, the choice of Applet in web simulation must be justified depending on the objective of the simulation works.

Conclusion

In this thesis, we have chosen the remote web-based CGI approach to implement Websim. The main reason for this is due to the popularity of C/C++ simulation engine. With this approach, we could reuse the existing simulation engine without much modification into the existing codes. The CGI program can directly execute and access the C/C++ program file. C/C++ programs are also well-known for their reliability, speed and small binary file size. The second reason is that many standard HTTP web servers still has not supported Java Data Server, whereas CGI is supported by almost all existing commercial and non-commercial web server, and can be configured with ease.

Simulation Languages

Simulation models are very often modelled using simulation languages. There are two categories of simulation language: general purpose and special purpose. In general purpose languages, there is no need to learn new set of syntaxes and semantics in order to use them. Languages which fall in this category include BASIC, Pascal, FORTRAN, C, C++, and Java. However, the using this general language approach requires modellers to construct all elements of simulation components from scratch. If available, there might be software libraries with which these elements can support a higher level of abstraction. Languages as such, may not be able to optimised for performance and the domains for which this language can apply are limited to simplistic model.

On the other hand, special purpose simulation language is designed for some specific application domains. Modellers may find this approach better defined with lesser overhead: it is not necessary to customize from a general language. A higher level of reusability in simulation objects can be achieved, as specialized components are provided for particular domain. reusability issue from software engineering perspective does not necessitate details for general re-use because specialized attention has been given and constructed for domain re-use, as such performance issues are immediately addressed. A more complex system for simulation can be constructed through special purpose language as these intrinsically provide less sophisticated language construct for mapping requirements.

We briefly discuss here the three most popular general purpose simulation language, namely C, C++ and Java.

C

C is one of the most widespread language available and a general purpose language in simulation. Its popularity has resulted in a large market for C-oriented tools, utilities and libraries. As a language with weak typing, errors are elusive and not usually capture until runtime. This language required additional add-on such as libraries for generating graphical environment and for simulation environment. By the same token, random number generators, process scheduling and synchronization must be complemented by the operation system or with simulation libraries. One of the advantages of using this language is its portability. Many development platforms support C language, as long it is an ANSI standard. Cross compilation for particular platform is merely through the recompilation process. The major deficiency of the language is it is not capable to support encapsulation of entities. It supports abstraction of entities to an extent to which only appeal to programmers who are familiar with this language. This is a general purpose language, which allows many C programmers to construct simulation with lower learning curves.

C language supports command line arguments. This feature is very important as the design and implementation of Websim, as we rely greatly on this feature. For example, we could give the command

```
sample_program data1 data2
```

to execute a program called `sample_program` that will will receive two inputs, the data1 and data2. C language provides a connection to the arguments on the command line. Typically, to use the connection, one would code

```
#include <stdio.h>
int main (int argc, char *argv[] )
{ ......
```

The parameter `argc` stands for "argument count". Its value is the number of arguments in the command line that was used to execute the program. The parameter `argv` stands for "argument variable". It is an array of pointers to char. Such an array can be thought of as an array of strings. The successive elements of the array point to successive words in the command line that was used to execute the program. Thus `argv[0]` is a pointer to the name of the command itself.

As an example of how this facility is used, suppose that we have written our program and have put the executable code in the file `sample_program`. The intent of the command line

```
sample_program data1 data2
```

is to invoke the program `sample_program` with the two input values `data1` and `data2` as command line arguments. In our program the three words on the command line will be accessible through the three pointers `argv[0], argv[1]` and `argv[2]`.

C++

Prior to the introduction of C++, C is commonly adopted as the simulation language due to its versatility and portability. C++ is an extension of C language, which enhances and improves on the shortcomings of C with its object-oriented features. C++ is a general purpose programming language with a bias towards systems programming that has the following characteristics [Stroustrup, 2000]:

- is a better C
- supports data abstraction
- supports object-oriented programming
- supports generic programming.

Java

Java is a good choice for web-based simulation language. Java is an objected oriented language which produces machine independent “applets”. The applets are in turn executed by Java-enabled web browser in ways similar to conventional program. Java is fundamentally different in that it allows applets downloaded from a web server to be executed locally on a browser.

The strongest reason for writing simulations in Java language is its ability to graphically display the states of the system as the simulation progresses. From the user point of view, the simulation process is easily visualized, while from the developer point of view, it helps in debugging code and result verification. Besides that, object reuse is made possible using Java. For instance, other people can download a copy of the developed Java class by simply clicking on a web link. This can then be included into the user program and run. As opposed to the traditional simulation language written in Simulator or C++, this approach does not require the Java code to be recompiled. Simulation models built using Java are generally flexible, easy accessible and provide network support (an integral part of Java language).

Current Java implementations compile down to an intermediate byte code, which is interpreted and therefore runs slower than other languages such as C/C++. Thus the main disadvantage of using Java is longer simulation times. [McNab and Howell, 1996] made a comparison between the two languages by writing a simulation that contains two entities which pass 2000 messages between them. The result shows that C++ run about 8 times faster than Java:

Table 1: Comparison between the performance of C++ and Java

Platform	**Average Execution time over 5 runs** (milliseconds)
Solaris C++	1538
Solaris Stand-alone using Java	12910
(Browser)Solaris Netscape using Java	11214
(Browser)Windows NT Netscape using Java	9341

Conclusion

Overall, simulation languages based on C/C++ have an edge over others with its code efficiency resulting in superior performance in simulation run. However, models built using this language generally requires recompilation of simulation libraries or code when ported to other platforms. In this thesis, we have chosen C/C++ as the simulation language for the following reasons:

(i) **Code efficiency** and **high speed execution run**. These two features are critical issues when performing web-based simulation. This is because the simulation engine resides on the server side and often requires very complex processing, hence a high speed simulation execution will result in the server having less load.

(ii) **Popularity.** C/C++ are very popular, especially among researchers in simulation fields. In the university, C/C++ are taught to students from almost all disciplines.

(iii) **Smaller File Size**. The executable binary file compiled in C/C++ are generally smaller in size. In Websim, size does matter as the file would be uploaded to the web server.

Web-based Simulation Technologies

In this section we shall describe commonly used technologies and tools to achieve web-based simulation, with particular emphasis on the technologies applied in the Websim project.

Common Gateway Interface

Transparency of network heterogeneity is among the key features of web based-simulation technology. Interoperability of different networks is achieved through well-defined, standardized protocols such as HTTP and Common Gateway Interface (CGI). The HTTP protocol defines a uniform information transport mechanism, while CGI establishes the interface between a web server and arbitrary program executing on the same machine [Ferscha and Richter 1997].

The CGI is a standard way for a Web server to pass a Web user's request to an application program and to receive data back to forward to the user. When the user requests a Web page (for example, by clicking on a highlighted word or entering a Web site address), the server sends back the requested page. However, when a user fills out a form on a Web page and sends it in, it usually needs to be processed by an application program. The Web server typically passes the form information to a small application program that processes the data and may send back a confirmation message. This method or convention for passing data back and forth between the server and the application is called the common gateway interface (CGI). These technologies are known collectively as server-side solution because the processing occurs on the Web server. CGI is part of the Web's HTTP protocol.

For example, if we were to create a Web site and want a CGI application to get control, we must specify the name of the application in the URL that you code in an HTML file. This URL can be specified as part of the FORMS tags if we are creating a form. For example, we might code:

```
                    <FORM METHOD=POST
    ACTION=http://cc.eng.upm.edu.my/cgi-bin/Websim.cgi>
```

and the server at "cc.eng.upm.edu.my" would pass control to the CGI application called "Websim.cgi" to record the entered data and return a confirmation message. The common gateway interface provides a consistent way for data to be passed from the user's request to the application program and back to the user. This means that the person who writes the application program can makes sure it gets used no matter which operating system the server uses (PC, Macintosh, UNIX, OS/390, or others).

It is a simple a basic way for information to be passed from the Web server on user’s request to the application program and back again. Because the interface is consistent, a programmer can write a CGI application in a number of different languages. The most popular languages for CGI applications are: C, C++, Java, and Perl. In this project, we have chosen Perl as the CGI language. Perl is regarded as the best choice for developing CGI programs because it has good text manipulation facilities and it also handles binary files. One problem with CGI is that each time a CGI script is executed, a new process is started. For busy Web sites, this can slow down the server noticeably. A more efficient solution, but one that it is also more difficult to implement, is to use the server's API, such as ISAPI or NSAPI. Another increasingly popular solution is to use Java servlets. [Whatis, 2000]

Practical Extraction and Report Language (PERL)

Perl is a programming language specially designed for processing text. Because of its strong text processing abilities, Perl has become one of the most popular languages for writing CGI scripts. Larry Wall, the creator of Perl, has this to say, "Perl is a language optimized for scanning arbitrary text files, extracting information from those text files, and printing reports based on that information. It's also a good language for many system management tasks. The language is intended to be practical (easy to use, efficient, complete) rather than beautiful (tiny, elegant, minimal)". This made Perl the suitable choice for Websim as this project require many writing and extracting of datas in and out from files.

Perl combines some of the best features of C, sed, awk, and sh. People familiar with the languages should have minimum difficulty with Perl. (Language historians will also note some vestiges of csh, Pascal, and even BASIC-PLUS.) Expression syntax corresponds quite closely to C expression syntax. Unlike most Unix utilities, Perl does not arbitrarily limit the size of the data if we have the memory, Perl can slurp in the whole file as a single string. Recursion is of unlimited depth, and the tables used by hashes (previously called ``associative arrays") grow as necessary to prevent degraded performance. Perl uses sophisticated pattern matching techniques to scan large amounts of data very quickly. Although optimized for scanning text, Perl can also deal with binary data, and can make dbm files look like hashes. Perl scripts are safer than C programs through a dataflow tracing mechanism which prevents many stupid security holes. [O'Reilly, 2000]

Perl is the most popular language for writing CGI programs which are invoked by the Web server when a user makes a request. A typical CGI program might process the contents of a form submitted to the server, and generate a customized reply. CGI.pm is the module most developers use to build Web applications with Perl. Written by Lincoln Stein and it can be retrieved from CPAN or directly from Lincoln's site. The current documentation is also available at the CGI.PM homepage [Stein, 2000].

Javascript

Javascript is a lightweight interpreted programming language with object-oriented capabilities. The general-purpose core of the language has been embedded in Netscape Navigator, Internet Explorer and other web browsers and embellished for web programming with the addition of objects that represent the web browser window and its contents. Javascript allows executable content to be included in web pages – it means that a web page need no longer be static HTML, it can include programs that interact with the user, control the browser and dynamically create HTML content.

In Websim, Javascript is used because of its ability to interact with HTML forms. The advantages of using Javascript here is that we would be able to perform verification of a form before it is submitted. If Javascript is able to perform all necessary error checking of a user's input, the CGI script on the server side can be much simpler and, more importantly, there is no round trip to the server to detect and inform the user of the errors. Javascript can also perform preprocessing of input data, which can reduce the amount of data that must be transmitted to the server. In some cases, client-side Javascript can eliminate the need for CGI scripts on the server altogether. On the other hand, Javascript and CGI work well together. For example, a CGI program can dynamically create Javascript code on the fly, just as it dynamically creates HTML. [Flanagan, 1998]

Java RMI

A Remote Method Invocation (RMI) capability has been included with the Java Development Kit (JDK) release 1.1 (Sun 1997). Remote Method Invocation is the object-oriented analogue of the traditional Remote Procedure Call (RPC) for distributed computation. In the Java distributed object model, a remote object is one whose methods can be invoked from another Java Virtual Machine, potentially on a different host. An object of this type is described by one or more remote interfaces, which are Java interfaces that declare the methods of the remote object. All remote interfaces extend, either directly or indirectly, the interface `java.rmi.Remote`.

A method in a remote interface must: (1) declare `java.rmi.remoteException` in its throws clause (in addition to any application-specific exceptions), and (2) declare remote objects passed as an argument or return value as the remote interface instance, not the class of the actual remote object, which is known as the implementation class. Architecturally, the RMI system consists of three layers:

(i) The stub/skeleton layer - client-side stubs (proxies) and server-side skeletons.

(ii) The remote reference layer - remote reference behavior (such as object invocation).

(iii) The transport layer - connection set up and management, and remote object tracking.

The relationship between these layers is noted in Figure 4.

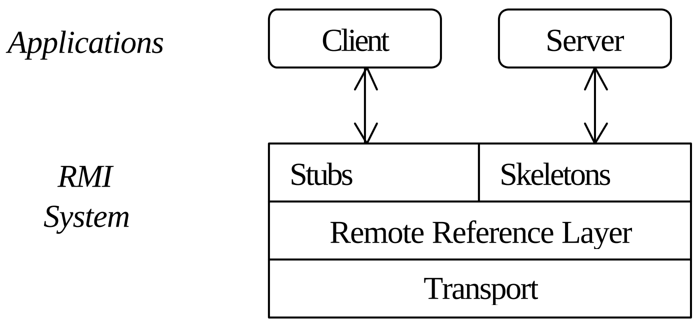


Figure 4: Remote Methods Invocation [Sun, 1997]

In this context, a server is any object whose methods are invoked remotely; while a client is an object that invokes a remote method. Thus, it is easy to envision applications of RMI (including distributed simulation) in which many clients and servers exist. Furthermore, a given object may - and commonly will - function as both a server and a client. Such an object will both invoke remote methods and have its methods invoked remotely.

A client invoking a method on a remote server object actually makes use of a stub (or proxy) for the remote object. The stub, an implementation of the remote interfaces of the remote object, forwards invocation requests to the server via the remote reference layer. The skeleton for a remote object makes an up-call to the remote object implementation which carries out the actual method call. The return call is sent back through the skeleton, remote reference layer, and transport on the server side, and then up through the transport, remote reference layer, and stub on the client side. Stubs and skeletons handle all marshaling for arguments and return values.

Also notable in the Java RMI: The remote interface must be public; the implementation class must create and install a security manager (even for Java applications); non remote arguments to, and results from, a remote method invocation are passed by copy rather than reference.

One of the primary advantages of these packages is that they permit network-based simulation models to be developed using established conceptual frameworks. [Ernest et al, 1997] describes work being undertaken to extend the Simjava package to utilize the Remote Method Invocation (RMI) facilities of JDK 1.1. This extension enables the construction of distributed, web-based simulations using Simjava.

Plotting Tools

Gnuplot is a command-line driven interactive function plotting utility for UNIX, MSDOS, and VMS platforms. It was originally intended as a graphical program which would allow scientists and students to visualize mathematical functions and data. Gnuplot supports many different types of terminals, plotters, and printers (including many color devices, and pseudo-devices like LaTeX) and is able to generate a GIF file too.

The software is copyrighted but freely distributed. Gnuplot handles both curves (2 dimensions) and surfaces (3 dimensions). Surfaces can be plotted as a mesh fitting the specified function, floating in the 3-d coordinate space, or as a contour plot on the x-y plane. For 2-d plots, there are also many plot styles, including lines, points, lines with points, error bars, and impulses (crude bar graphs). Graphs may be labeled with arbitrary labels and arrows, axes labels, a title, date and time, and a key. The interface includes command-line editing and history on most platforms. [Gnuplot, 2000]

In this thesis, Gnuplot is used to generate the final simulation results as a graph image. First, it would read the results in text files format from the simulation program and output the plotted graph as GIF file. The GIF finally would be embedded into an HTML file and displayed to the user.

Conclusion

From the discussion and reviews, it is clear that a large amount of work has been carried out over the years to make simulation over the web possible. Compared to the traditional method of performing simulation on a standalone environment, web-based simulation has proven that it provides many advantages over the traditional methods such as platform-independent, flexibility, multi-users support, etc.

Web-based simulation programs generally fall into two categories. Simulation programs that can be accessed remotely through web browsers and form-based CGI scripts comprise the first category. Typically, these simulations allow the user to tailor (via the forms interface) model execution parameters such as mean service times and arrival rates, number of model replications, and so on. A single copy of the simulation runs on a server and passes the results of model execution to the invoking client. The second category of web-based simulation programs represents a variation of the first, but with the added feature of code mobility afforded by such network programming languages as Java. Here, the simulation executes on the client rather than the server. Java-based simulation-support libraries are emerging that permit the creation of simulation programs as Java applications and applets.

Nonetheless, there are some common desirable features expected from an ideal integrated web-based simulation environment, including fully platform independent, multi users support, user-friendly interface, easy setup, fast, support of animation, and secured.

Websim belongs to the first category where simulation programs are accessed remotely through web browsers and forms-based CGI scripts. This thesis is the start of a fully integrated web-based simulation environment initiative. It basically provides general solutions for the execution of simulation over the web. Among the features provided by Websim are: platform independent, multi users support, user-friendly interface and easy setting up. Since security is a crucial and very important aspect in any server-side processing, some measures have been incorporated to make Websim more secured. This is further discussed in Security Issues section (page 65).

CHAPTER III

WEBSIM SYSTEM DESIGN AND ARCHITECTURE

Introduction

Websim shows how one could extend the capability of simulation tools into the web. It is designed in order to provide distributed and platform independent features to multiple users. For instance, a user could perform the simulation works at home using the simulator located at his office, while another colleague at the office watches the results shown on the computer screen via a web browser.

The flow of processes in Websim is as follows. First there must be an executable simulation program developed using C/C++ Language. The administrator would upload the executable file into Websim, and a web-interface for the file would be created. The web-interface would allow end-user to perform the simulation processes, using the web browser to receive input data. After the completion of the simulation processes, the results generated from the program would be translated into an image file using the Gnuplot graph plotting program. Finally, the image file would be embedded into an HTML file for the user to view. The entire flow of this system is controlled by a central CGI script, which resides on a web server.

This chapter describes the design and architecture of Websim, which is based on a client-server architecture. It consists of several components residing on the server as well as client side, which allows the creation of a web-based simulation interface for a standalone simulation program.

Client Server Architecture

Websim is developed based on client-server architecture in which application processing is divided between client and server, and a central web server coordinates the access to the local computational resources. Client-server architecture refers to a LAN based computing environment in which a central server or engine performs all command sent to it from client workstations, and the application programs on each client concentrates on user interface functions. The client-server interaction and its components are shown in Figure 5.

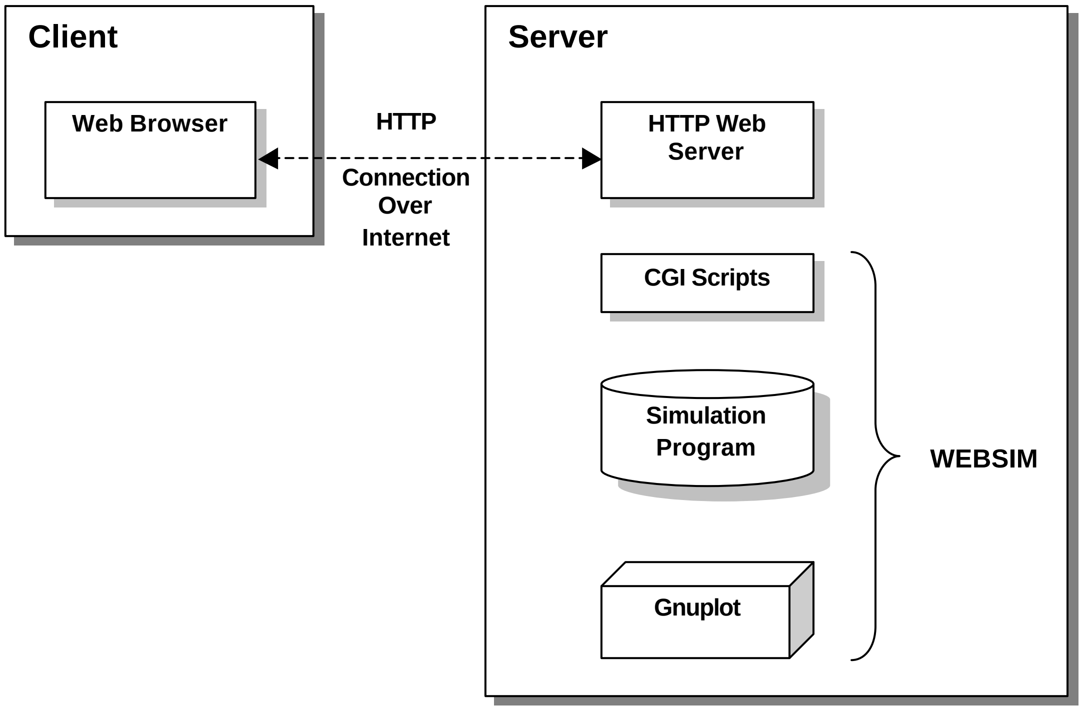


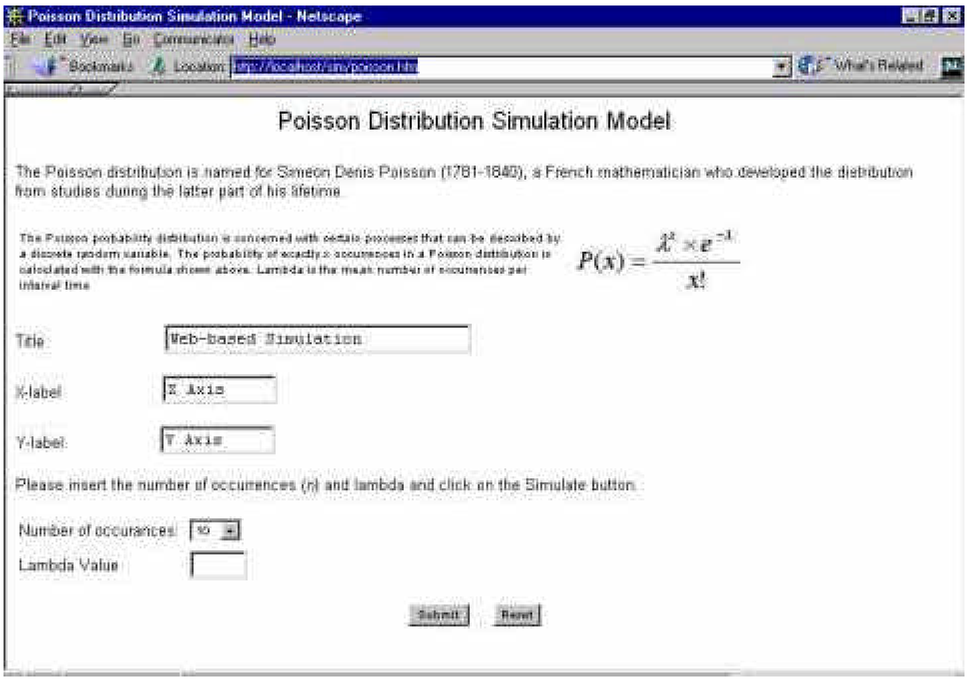
Figure 5: Client-server interaction in Websim

Web Browser

Referring to Figure 6, the single component on the client side is the web browser. It is used to access the Websim services provided on the server. When the browser user enters file requests by either "opening" a Web file (typing in a Uniform Resource Locator or URL) or clicking on a hypertext link, the browser builds an HTTP request and sends it to the Internet Protocol address indicated by the URL. The HTTP daemon in the destination server machine receives the request and, after any necessary processing, the requested file is returned.

Whenever the URL of Websim is referred, the web browser would display an input form as explained in the next section. It is via this input form that the user may interact with the server. Web browsers make the Internet a more user-friendly environment by integrating all those resources into a single tool that shorten the learning curve. Besides, it may also support the incorporation of user interaction and animation into the simulation tools.

The only component of data collection in Websim, the input form is implemented in HTML and CGI. The main advantage of HTML forms is that we can use them to create a front end for numerous gateways (such as databases or other information servers) that can be accessed by any client without worry about platform dependency. As shown in Figure 6, the user would enter the data onto a text field on the input form. When the submit button is clicked, the data is encoded and sent to the CGI program which is associated with the input form.


Poisson Distribution Simulation Model - Netscape
File Edit View Go Communicator Help
Bookmarks
Location: http://localhost/sim/poisson.htm
What's Related

Poisson Distribution Simulation Model

The Poisson distribution is named for Simeon Denis Poisson (1781-1840), a French mathematician who developed the distribution from studies during the latter part of his lifetime.

The Poisson probability distribution is concerned with certain processes that can be described by a discrete random variable. The probability of exactly x occurrences in a Poisson distribution is calculated with the formula shown above. Lambda is the mean number of occurrences per interval time.

$$P(x) = \frac{\lambda^x \times e^{-\lambda}}{x!}$$

Title	Web-based Simulation
X-label	X Axis
Y-label	Y Axis

Please insert the number of occurrences (n) and lambda and click on the Simulate button.

Number of occurances	10
Lambda Value	

Figure 6: Screen shot from the input form

The core server site components of the *Websim* architecture are HTTP Server, Controlling CGI Script, Simulation Program and Gnuplot plotting program.

HTTP Web Server

The HTTP Web server is a computer program that serves requested HTML pages or files from the Web client. Web server machine contains, in addition to the HTML and other files it can serve, an HTTP daemon, a program that is designed to wait for HTTP requests and handle them when they arrive. In our case, the web client is the simulation user, who uses a standard web browser. There are many Web server software applications, including public domain software from NCSA and Apache, and commercial packages from Microsoft, Netscape and others. We have tested Websim on two packages: (1) OmniHTTPd by Omnicron Technologies Corporation [Omnicron, 2000]; and (2) Microsoft Internet Information Server (IIS) by Microsoft Corporation [IIS, 2000]. Both are considered fast and compact for the Windows platform. In addition to Standard CGI support, the servers support advanced features such as Keep-Alive connections, table auto-indexing and server-side includes. But IIS outperforms OmniHTTPd in terms of providing server security and protection, as it has many integrated features with the Windows NT operating system which we are running on.

Controlling CGI Script

The execution of the simulation is controlled by several central CGI-scripts, residing on the web server. These scripts, written in PERL (version 5), are the most important part of the system. They act as the brain to the entire system. We will further discuss the functionality and operation of the scripts in the coming section. The scripts act as a central controller of the overall system processes and provide the Websim services such as:

- User authorization service.
- Simulation Program Upload service.
- Web interface generation service.
- Program execution service.
- Graph Plotting service.
- Image Viewer service
- Site management service.

These services are discussed in next section.

Simulation Program

The simulation program refers to any application written in C/C++ language to simulate a certain simulation model. It must first be compiled using any standard ANSI C/C++ compiler and made into an executable file by the user before uploading to the Websim server. In order to achieve full compatibility with Websim, we have set three criteria that the Websim administrator must follow when developing his simulation program:

(i) All input and output variables must be assigned command line arguments

(ii) Name of the output file must be of the name “output.txt”

(iii) The format for the simulation result file “output.txt” must be arranged in columns.

To make it easier for the user to see, we have designed a standard template for the C/C++ simulation program, as shown in Figure 7:

```
// Websim Template version 1.0

#include <stdio.h>
#include <stdlib.h>

/*
main program (assuming)
Two input parameter:  a and b
Two output parameter: text files
*/

int main (int argc, char *argv[])
{     int a, b;
      FILE *ofp;

//    to convert a string to integer or other data types
//    next is to pass the values for processing
      a = atoi(argv[1]);
      b = atoi(argv[2]);

// opening files for output, using names like output.txt
// if there is > 1 file, name the second file as output2.txt
and so on
      ofp1 = fopen (“output.txt”, "w");
      ofp2 = fopen (“output2.txt”, "w");


// print the results of simulation into the output file using
2 or
// more columns
      fprintf (ofp1, ……………);
      fprintf (ofp2, ……………);

//close the files
      fclose (ofp1);
      fclose (ofp2);
      return 0;
}
```

Figure 7: C/C++ Simulation Program Template

One simple program is shown here. This program receives two input parameters (*x* and *y*) and produce one output text file (Figure 8). The result of the computation would be output into a text file with n number of rows and two columns. The first column indicate the values for the X axis and the second column for the Y axis.

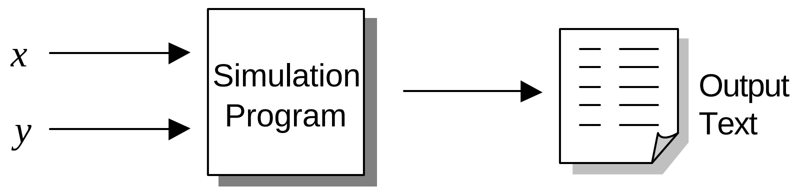


Figure 8: Input and Output Parameter of Simulation System

Gnuplot

Gnuplot is a software application suited for graphing simple numerical information [GNUPlot,2000]. The latest version for MS-Windows 32 bit system is version 3.7. It has the ability to take raw data and create various types of graphs, including point and line graphs and histograms. Further explanation has been given in Plotting Tools section (page 25).

System Services

Websim provides an environment for users to create, run and maintain a web-based simulation project. This is made possible by defining a set of useful services which operate as a whole in the system. Websim provides six important services for Websim user, and each of the services is explained below.

User Authorization Service

Authorization service ensures that only authorized users can access Websim by requesting the user to enter his/her login name and password. The name and password will be verified based on a text file, which stores the names of all authorized users. There are two types of Websim user: normal user and administrator.

Normal users are allowed to run an existing web-based simulation project, view the output GIF images, and customize plotting properties. As for the administrator user, they have all the normal users accesses, plus two more privileges. Firstly they are allowed to create a new web-simulation project, hence has the permission to upload file(s) to the web server. Secondly, they may edit, modify or delete existing files.

Simulation Program Upload service

This upload service provides a step-by-step instruction on how to create a web-based simulation project on the fly. This service is exclusive for administrator users only. After naming the project, the administrator may upload the simulation program file (.exe) to the project folder. For the moment, we limit the maximum size of the file to 500 KB. We also provide a verification service to ensure that only file with a ".exe" extension be allowed to be uploaded.

Web Interface Generation Service

This service is part of the upload service, and is very important in providing the web-interface for the uploaded simulation file. After a file is uploaded, Websim will generate a HTML file, which acts as the cover page for the simulation program.

This HTML file will include the text fields for users to enter the simulation parameters to be passed to the simulation program.

Program Execution Service

This service allows the user to run the simulation program uploaded by the administrator earlier. This service provides an input form containing text fields where user can insert the data values to be processed by the simulation program. When the submit button is clicked, the CGI script associated with this service will call the simulation program and pass the values entered earlier on. Once the simulation is done, the result would be output into a GIF file. The user may choose to keep the file at the server or download it to the local computer.

Graph Plotting Service

This service allows the Websim user to customize the default output graph settings. Several properties such as the graph size, line types, show grid may be changed and set. Apart from that, the user may also plot his own output text files other than the default “output.txt” file. The user may preview the output graph based on the settings made earlier.

Image Viewer Service

These users may display all GIF images of simulation graphs in a chosen project directory. The images are displayed in a table format with their names shown at the bottom. The users may view or download them to their local hard disks.

Site Management Service

This service is for administrator use only. It performs most of the file operations required for web site maintenance such as edit, delete and upload files. For example, the administrator may use this service to edit and insert an image components to his simulation project web interface.

Websim Actors

An actor is someone or something that must interact with the system under development [Rational, 1997]. There are two types of Websim actors: the administrator and normal user. The administrator has the authority to create and manage a simulation project. They are granted with upload, change, delete and other administrative services. Meanwhile, normal users are only allowed to view and execute the simulation system. Figure 9 shows a use case diagram to visualize the relationships between two users and their services. As shown below, the administrator has two additional services compared to a normal user.

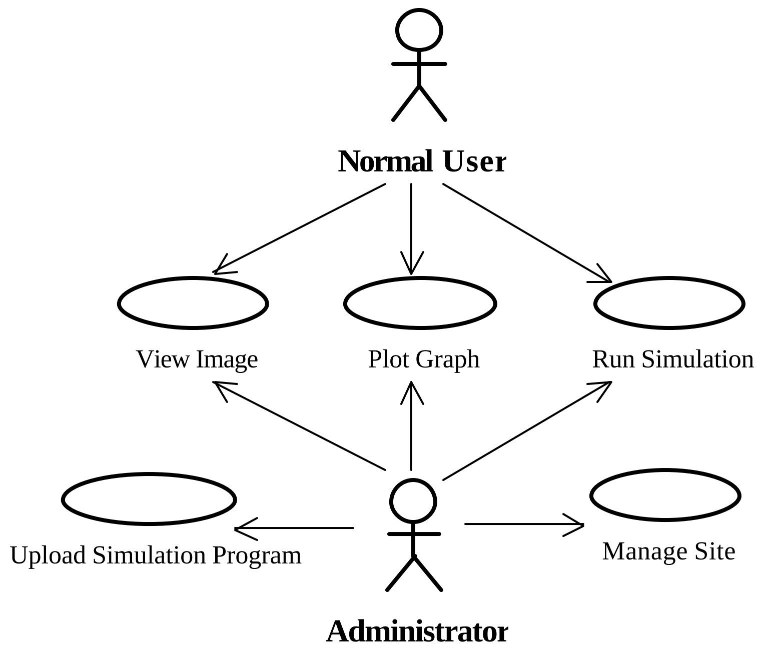


Figure 9: Relationship between Websim users and their services

Websim Logical Design

Logical design is the most crucial stage of software development cycle. It is at this stage where we elicit and structure all information requirements for Websim [Hoffer, et al, 1996]. Here we provide functional specifications of all data, forms, computer displays and processing rules for all aspects of the system.

The controlling CGI script is considered as the most important component of Websim. It is these scripts that provide all the services mentioned in previous section. These scripts are known as modules and each module is written to perform a specific task. Six modules have been developed using Perl and Javascript. There are three basic modules for both administrator and normal user. But the administrator has two additional modules, which are the 'Upload Wizard' and 'Site Manager' modules.

We provide also a main menu to integrate all the modules. Figure 10 shows the inter-relationships among the major components of Websim.

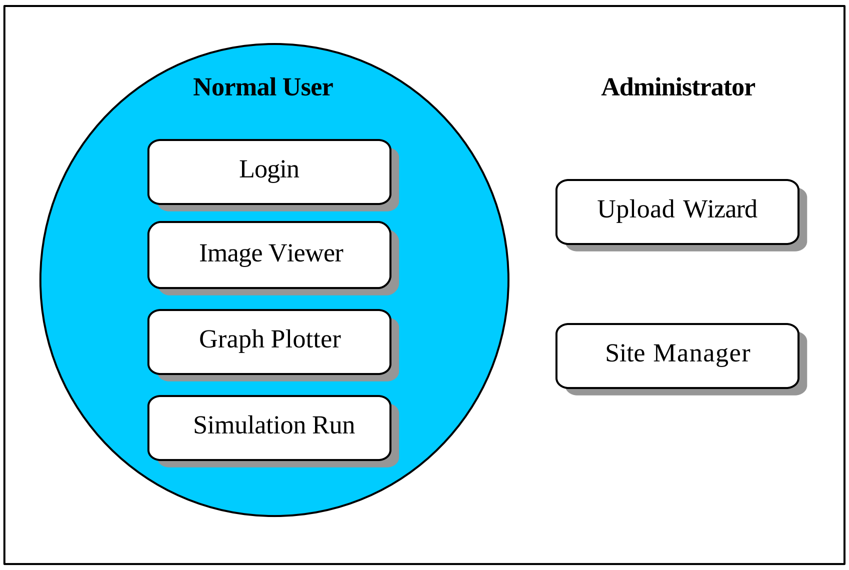


Figure 10: Websim Modules

To further explain the overall modules provided by Websim, we present here the overall sequences Websim users follow to interact with the system. Figure 11 shows a dialogue diagram, a formal method for designing and representing human-computer dialogues using box and line diagrams. It presents the sequence of interaction between a user and the system.

As an illustration, an administrator must gain access to Websim through a log-on procedure (item 0). If log-on is successful, a main menu is displayed that has five items (item 1). Once the administrator selects the Site Manager (item 3), control is transferred to the Site Manager page (item 3.1). Here, the administrator is presented with three options to edit, delete or upload file to the server (item 3.1.1). Once the administrator edit a file, the system will allow him to back up to select a different file, or return to the main menu (see bottom of item 3.1.1).

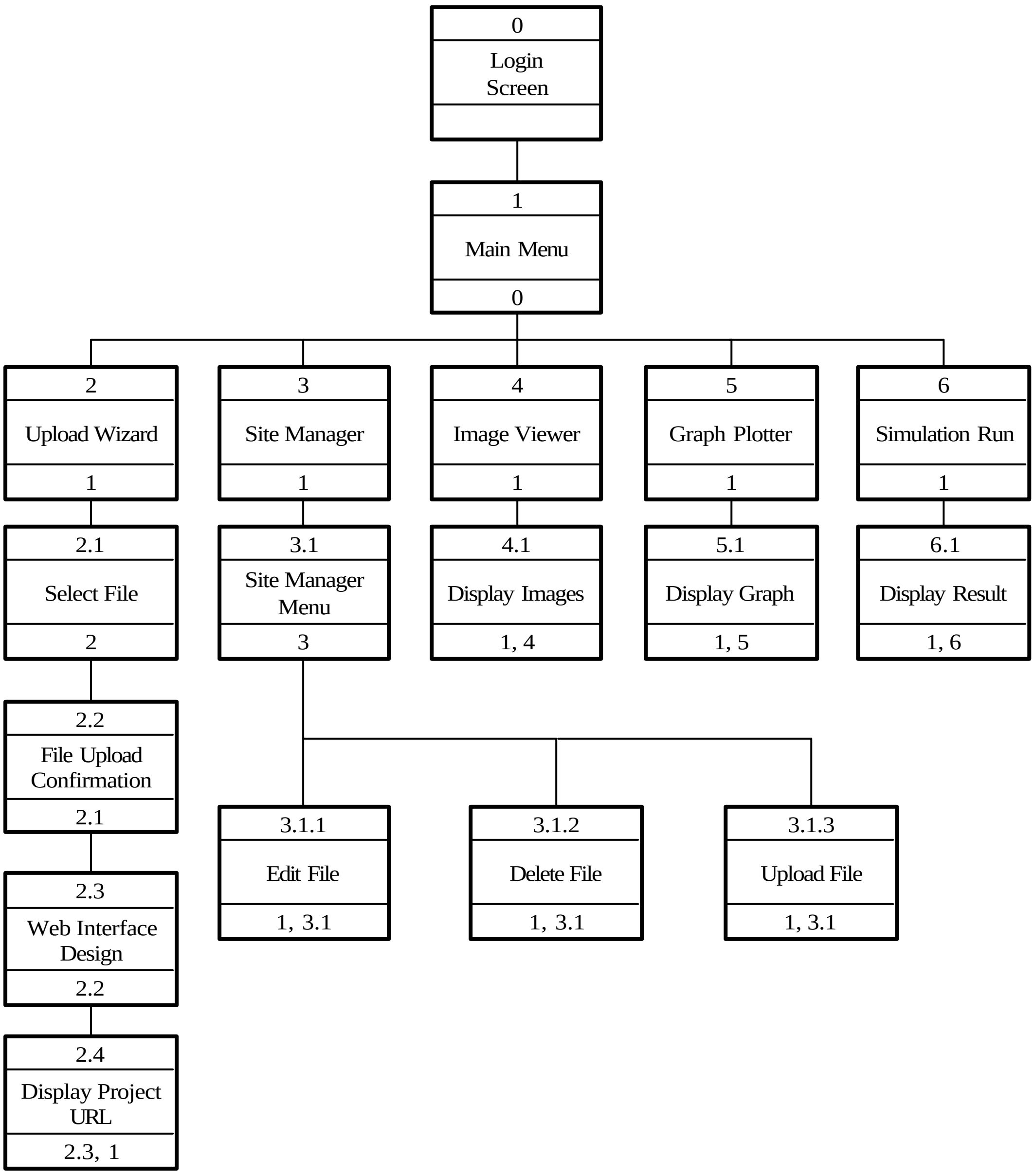


Figure 11: Dialogue Diagram showing sequence of interaction between Websim users and the system

In the next section, we describe each module by showing their data flow diagrams (DFD). DFD refers to a picture of the movement of data between external entities (e.g. Administrator) and the processes and data stores within the system. DFD also shows the processes that change or transform data, and how data come to be stored at specific locations. The diagrams shown are called level-1 diagram as they represent only the primary individual processes in Websim at the highest possible level.

Login Module

The DFD for this module is represented by Figure 12. Both Websim administrator and normal users may access this module. This module has two inputs: username and password. Once the username and password are entered, Websim will verify them by matching the name and password with datas contained inside a text file named “password.txt”. An error message will be displayed if the name and password entered are not found or does not match. If the password is correct, the administrator may choose to create a new project or open an existing project; while the ordinary user will be shown a page containing all the projects that he is allowed to access. After this process, the main menu for each user type will be displayed.

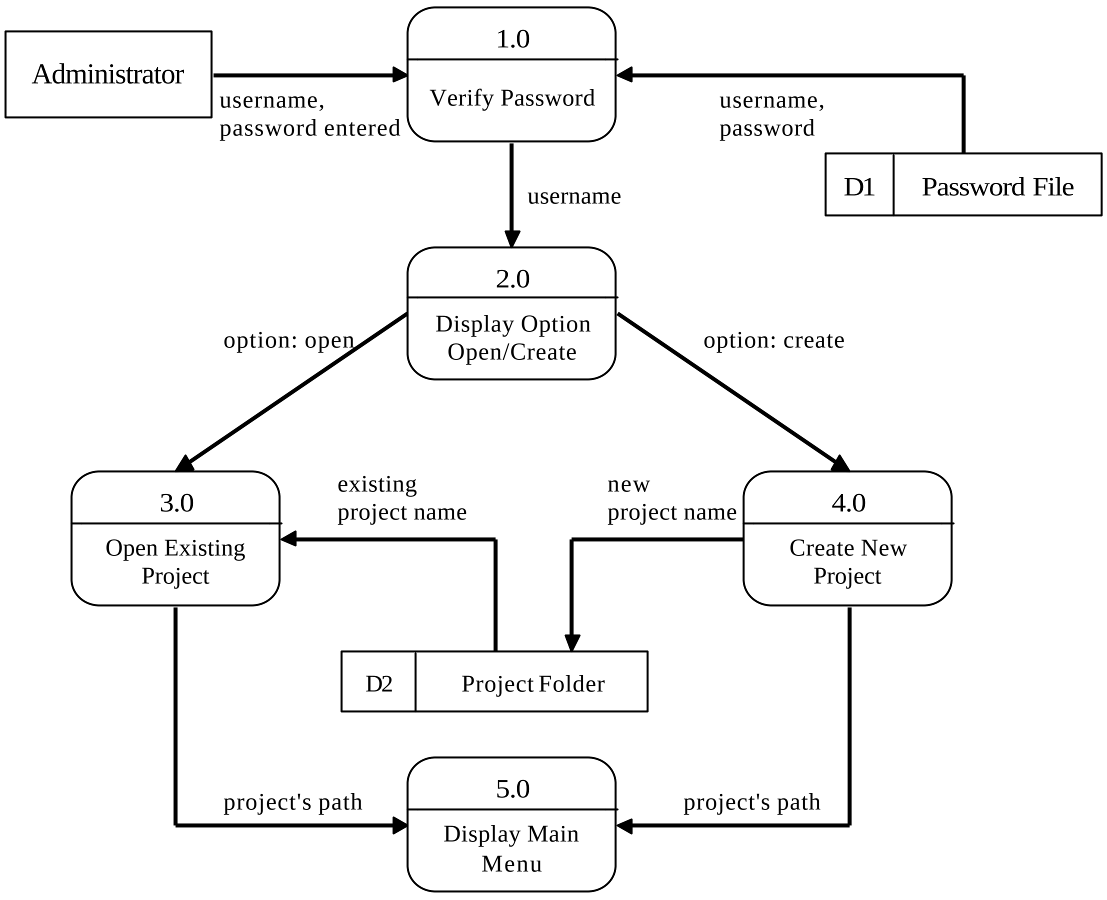


Figure 12: DFD of Login Module

Upload Wizard Module (Administrator only)

This module begins with the simulation program file select and the number of parameter input from the administrator. In the first process, labeled "1.0", we see that the data are processed. The results are two flows of data: (1) the number of parameter value is stored in cookies, (2) the filename is sent for validation process. Once the file is accepted, it will be stored in the server simulation project folder. Process "4.0" will read the data from cookies and create a HTML file, which acts as web-interface for the simulation file uploaded earlier. When this is done, URL for the simulation file will be displayed on the screen. The DFD for this module is shown in Figure 13.

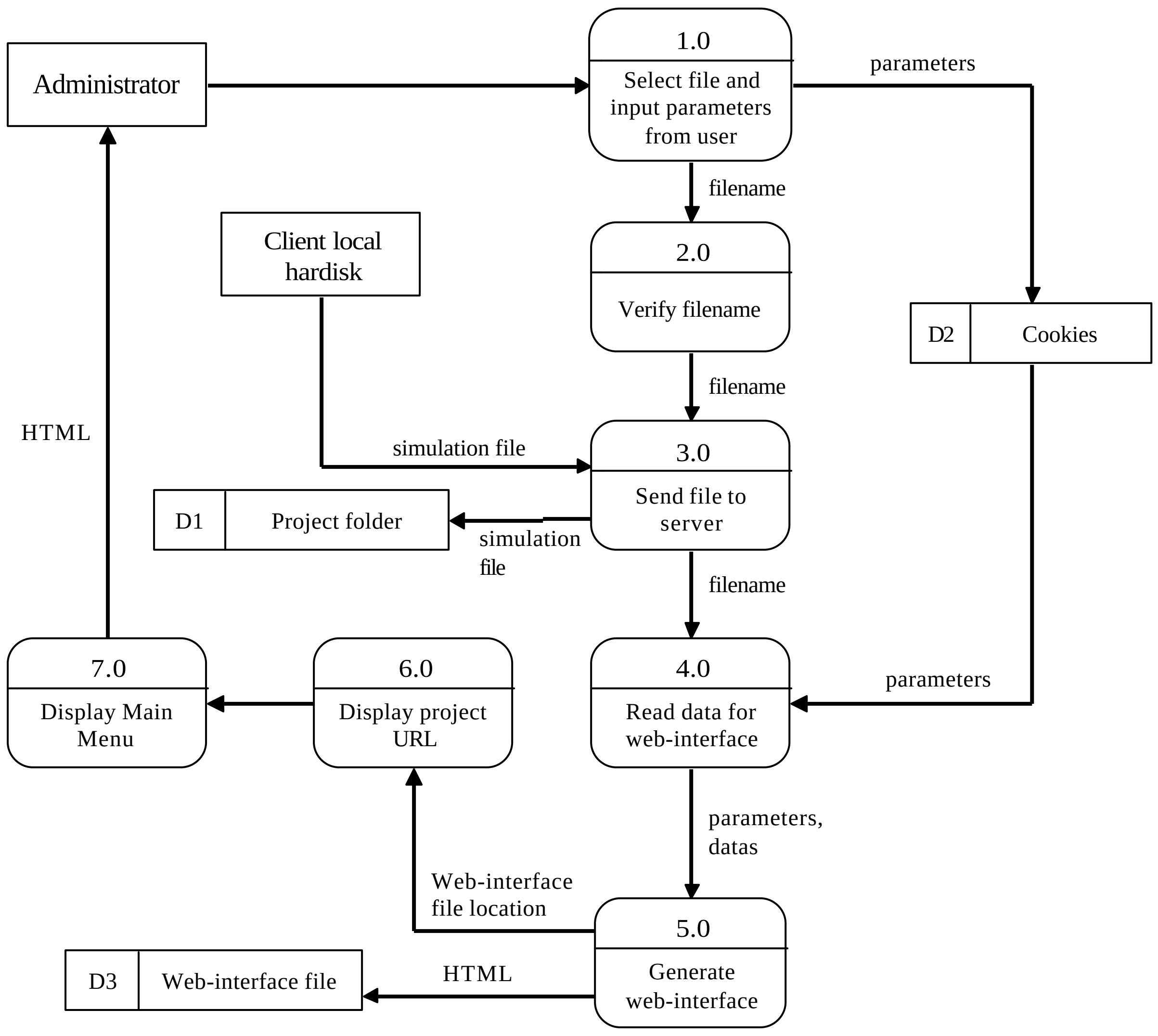


Figure 13: DFD of Upload Wizard Module

Site Manager Module (Administrator only)

This module is exclusive for administrator use only. They are edit, delete and upload options. Here we use three different DFDs to represent the three different options. In the case of delete file, process "1.0" requires the administrator to select a file from the project folder, and before the delete command is executed, process "2.0" will first ask for the user confirmation. The DFD for this delete option is shown in Figure 14.

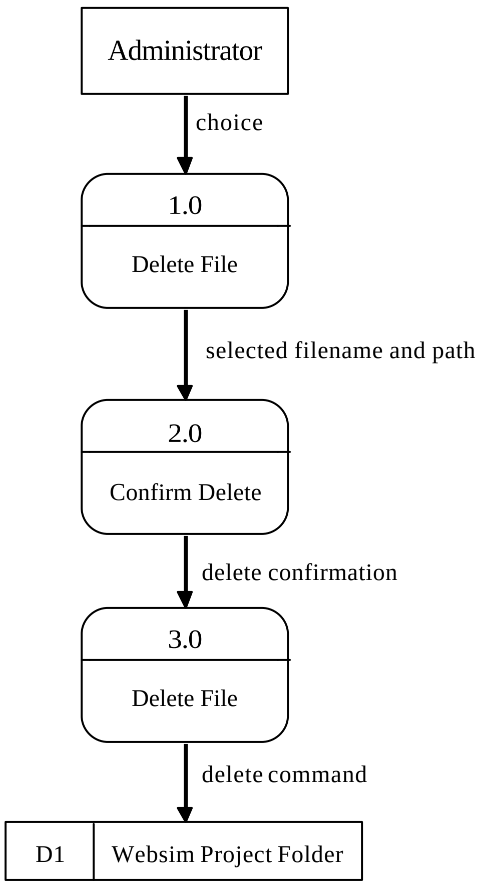


Figure 14: DFD of Site Manager Module - Delete Option

As for the edit option, process "1.0" select a file from the project folder and display its content on the screen for the user to edit. Process "2.0" receive the filename and open the filename from the Websim project folder, and display its content. The edited file finally is saved in the project folder using back the original name. The DFD for this edit option is shown in Figure 15.

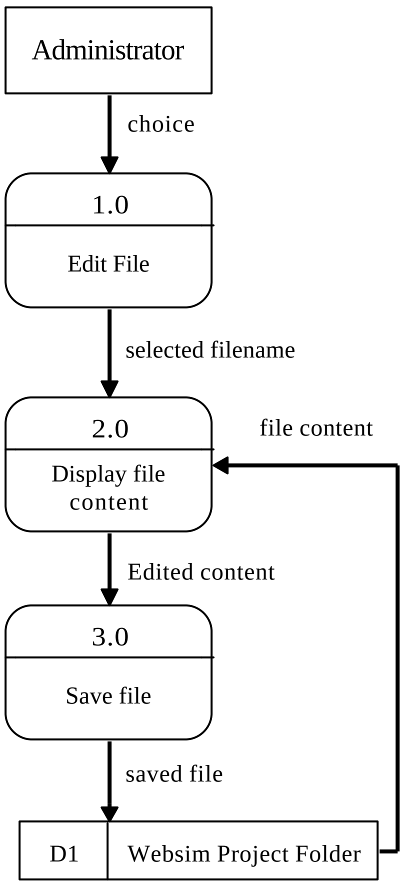


Figure 15: DFD of Site Manager Module - Edit Option

To upload a file using the upload option, first the administrator needs to select a file local hard disk as shown in process "2.0" in Figure 16. The filename will be verified in process "3.0" to ensure only certain file types are allowed to be processed. Once the filename is verified, process "4.0" will send the file from the local hard disk to the server, and store it in the project folder. The DFD for this upload option is shown in Figure 16.

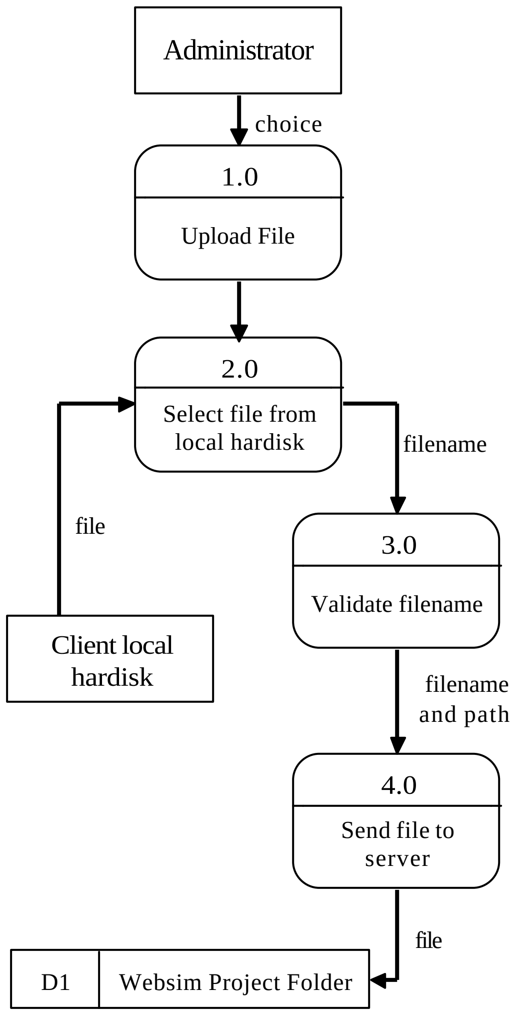


Figure 16: DFD of Site Manager Module - Upload Option

Image Viewer Module

This module allows both administrators and normal Websim users to view the simulation result files. When a user selects this choice, process "1.0" will read all GIF and JPEG files located in the project folder. Process "2.0" will then arrange all the image files and display them as HTML file format. The DFD for this module is shown in Figure 17.

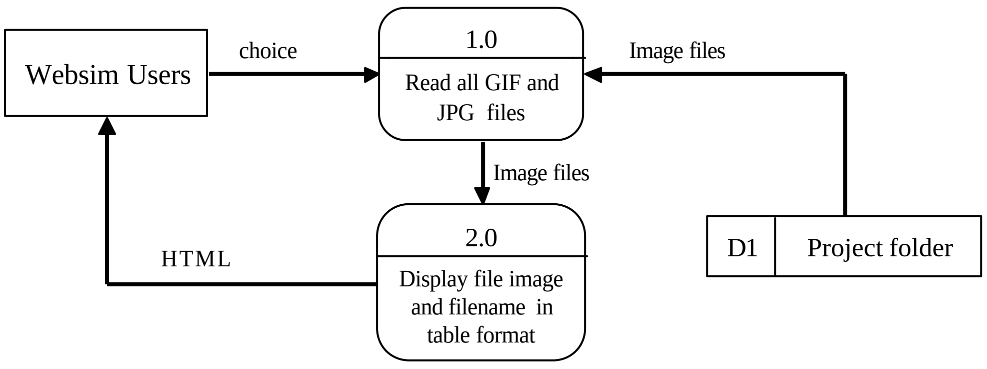

Websim Users
choice
1.0
Read all GIF and JPG files
Image files
Image files
2.0
Display file image and filename in table format
HTML
D1
Project folder

Figure 17: DFD of Image viewer

Plotting Module

The DFD for this module is shown in Figure 18. This module begins with the selection data file to plot, as in process "1.0". The data file from the project folder becomes the input data file for process "2.0", when it execute the simulation program (as shown in process "3.0"). The simulation program execution will produce one output text file, and process "4.0" will plot a graph using data from the output text file. Lastly, process "5.0" will take the output image from process "4.0" and embed into an HTML file, before sending it to Websim users.

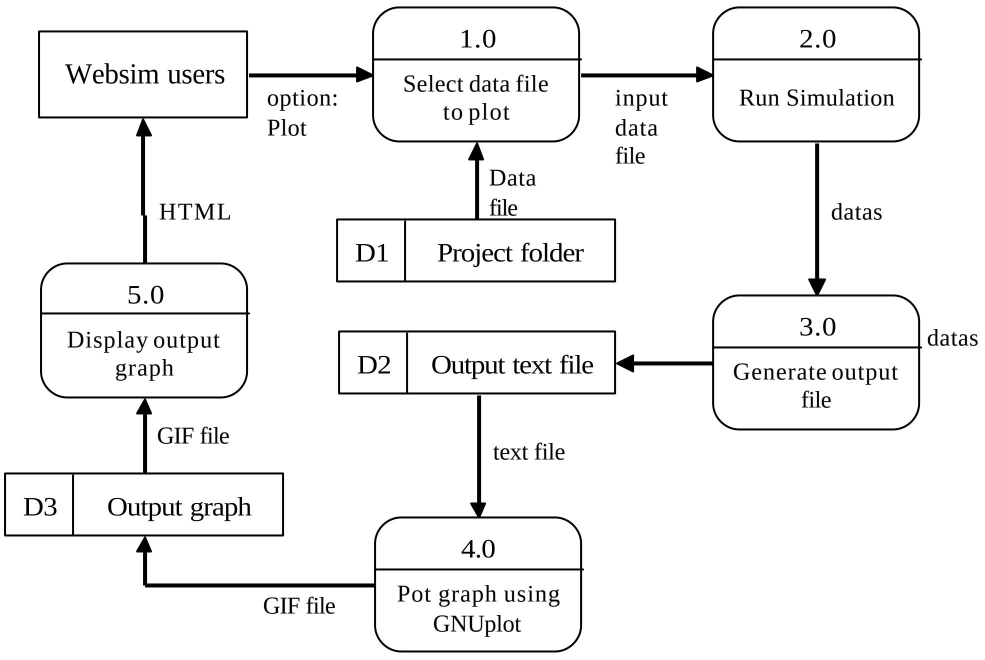


Figure 18: DFD of Plotting Module

Simulation Run Module

When this module is chosen, the first process displays the default page "index.htm" to the Websim users. The default page contains a list of HTML form tags, which allows users to input the simulation input data. Process "2.0" will process these input data and generate a text file. Process "3.0" calls the simulation program and produces an output text file. Process "5.0" calls the plotting program, plot the data in the output text file and output an GIF image file. Lastly, process "6.0" takes the output image and embed into an HTML file, before sending to the Websim users. The DFD for this module is shown in Figure 19.

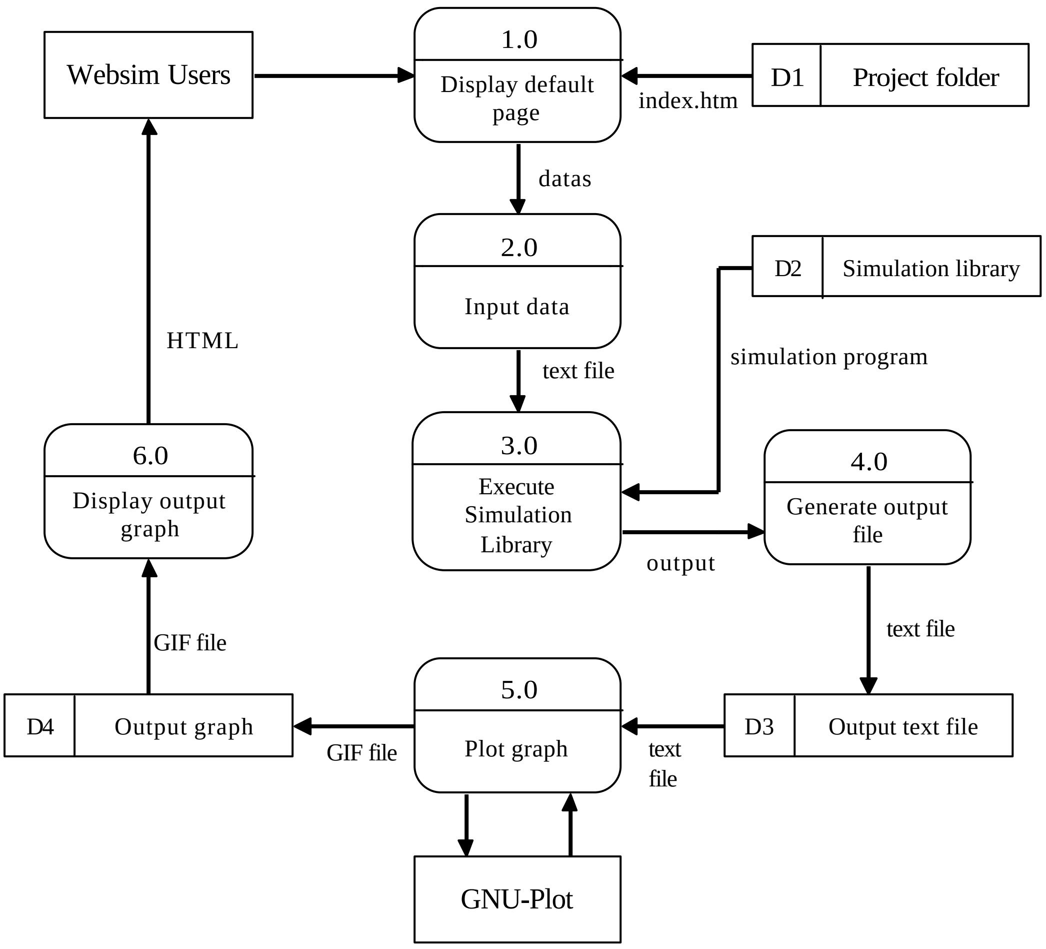


Figure 19: DFD of Simulation Run Module

Files Used in Websim

In this section we present the file system layout of Websim. Websim are made up of HTML, CGI scripts and text files, each positioned at a particular folder. Table 2 lists the file types used and Figure 20 shows the file system layout for Websim.

Table 2: Files used and their descriptions in Websim

File Type	Description
`index.htm`	Default HTML file displaying the welcome and login screen.
`Websim.htm`	Display all existing simulation projects for normal users.
`login.cgi`	Script which verify username and password, as explained in the login module.
`wizard files`	Five scripts providing the step-by-step upload wizard service, as explained in the upload wizard module.
`manager.cgi`	Provide the site manager service, as explained in the site manager module.
`gnuplot.cgi`	Provide the graph plotting service, as explained in the plotting module.
`image.cgi`	Provide the image viewer service, as explained in the image viewer module.
`main.cgi`	Display the main page for the user, providing the main menu where user may access all Websim services.
`sim.cgi`	Script that performs the processing of input data from user, and executing the simulation program.
`default.dat`	The data file used for GNU-Plot program, containing the default setting for the required output graph.
`passwd.txt`	Contain the username and password, which the login script refers to when verifying Websim users.

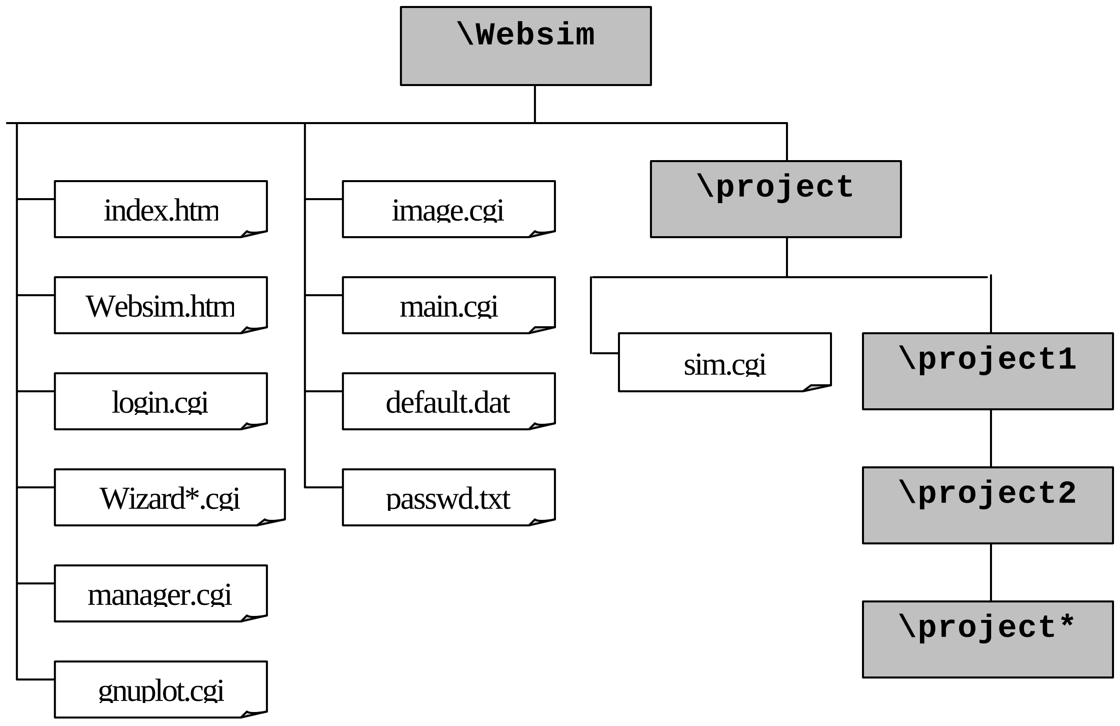


Figure 20: The layout of the directories and files for Websim