Developing modular software with reference to web hosting automation

BSc (Hons) Software Engineering

Karl Peter Hill Austin

Acknowledgements

I would like to thank Mike Parr, for help and advice throughout the project when I’ve needed it.

I would also like to thank KDA Web Services Ltd. For the use of a Linux server connected to the Internet for using as a remote client server in the system and for providing an account on another Linux server from which the control software could be ran.

Thanks are also due to Positive Software for answering questions on how their system works and to cPanel Inc. for doing the same.

Thank you to many customers of KDA Web Services Ltd. For helping with my research and for giving their valuable time and opinions on what they think of HSphere and of CPanel.

Abstract

The aim of this project was to develop ideas for creating modular and scalable software for applications relating to web hosting automation software. A small scale proof of concept deliverable demonstrating these ideas was produced, implementing the design decisions taken. The proof demonstrates the use of Object Oriented design, utilising the Factory Pattern in its implementation and also utilising XML and XSLT to help with the configuration and to aid the platform independence goals of the project.

Some problems were encountered along the way, but these tended to be quite trivial such as tabs in a configuration file stopping it being parsed by Bind and other similar issues.

The goals of the project have been met, the ideas developed are workable in a real world situation and could be used to develop a highly modular hosting automation system, or general client-server control system.

Keywords

internet hosting client-server modular

Contents

1. Investigation and analysis

1.1 Introduction

Since 1996 the Internet and more specifically the World Wide Web (WWW) have taken the world by storm. What once used to be the domain of scientists and academics is now accessed by millions of people from all walks of life. This huge increase in popularity has led to a large increase in the number of web sites online, which currently stands at over 39 million (Netcraft, 2003), up from just under75,000 in January 1996 (Netcraft, 1996). All of these web sites need a place to be stored on the Internet, usually on the web servers of many commercial web hosting providers.

The problem with such a large number of web sites is how to provision them and how to manage them in an efficient and user friendly manner. Not so many years ago, web sites were setup by hand, from the command line of a server, or using a collection of shell scripts written over a period of time by the system's admin, again, run from the command line. In recent times, it has become commonplace for web sites to be provisioned and managed by automated software systems, referred to as control panels. These systems usually provide an administrative and customer interface to control both the provisioning of user accounts and the customer management of a site and related items such as email.

As hosting providers come to rely ever more on automated control systems to take care of day-to-day tasks it is important that the software be able to grow with the company and meet the needs of their customers. If a company were to find that their control system cannot scale with their growth or cannot keep up with the latest technology then it could prove costly in terms of new and existing business losses. If the software a company choose cannot scale any further then they have no choice but to go through the expensive and time consuming task of moving between control systems – which is not adequately planned and prepared for can result in many problems for the company and customers alike, with the ultimate consequence being web site unavailability. So as you can see it is important for a company to find software that will grow with their business and be able to keep pace with the market in terms of new features that customers want and expect.

What I will be looking at in this report and demonstrating via a proof of concept are the tools that can be employed by web hosting companies to provision and manage web sites and looking at ways they can be improved from the perspective of scalability and expandability - Two key issues for any large provider of web hosting. I will be focusing my research on two of the most popular tools on the market for this task, HSphere from Positive Software and CPanel6 from cPanel Inc. The main focus of my research and development will be in to ways that modularity can be built into the software, to allow for easy addition of new features as well as the ability to expand existing features without having to have access to the existing source code, e.g. A third party developer could add a module to work with the Zeus Web Server, without having to edit any existing code in the system.

1.1.1. Project Specification and mark weighting

After reviewing the project, I believe it was important to modify the project specification to give more emphasis in to the research aspect of the project rather than to the deliverable, as the deliverable will only be a proof of concept (as outlined in the original specification) and the original specification placed an emphasis on the modularity of the system, but did not allocate enough time in to researching ways of achieving the aim and allocated more time to the deliverable than needed. Also after taking a brief look at two of the most popular solutions in the market place, I decided that more time should be allocated to understanding how they work so that I can better evaluate them and learn from how they work. To reflect the shift in focus, the marks breakdown has been adjusted accordingly as it placed too much emphasis on the deliverable when it was only ever planned to be a proof of concept. The revised specification is attached as Appendix A.

1.2. Requirements to host a web site

The basic requirements to get any web site up and running with a domain name are:

A Domain Name
Server Hardware and Software
An Internet Connection
Web Server Software
DNS Server Software

1.2.1 DNS Server

For the purposes of demonstrating the modularity and expandability of the design/coding ideas and techniques I shall demonstrate the use of two DNS servers, specifically BIND and DJBDNS, two competing open source products. The diagram below shows what happens when a user types a domain name in to their browser and the role DNS plays. In the case of the system I will be developing, the DNS server for the system fits in to the box, “Query each of the nameservers until you get an answer”.

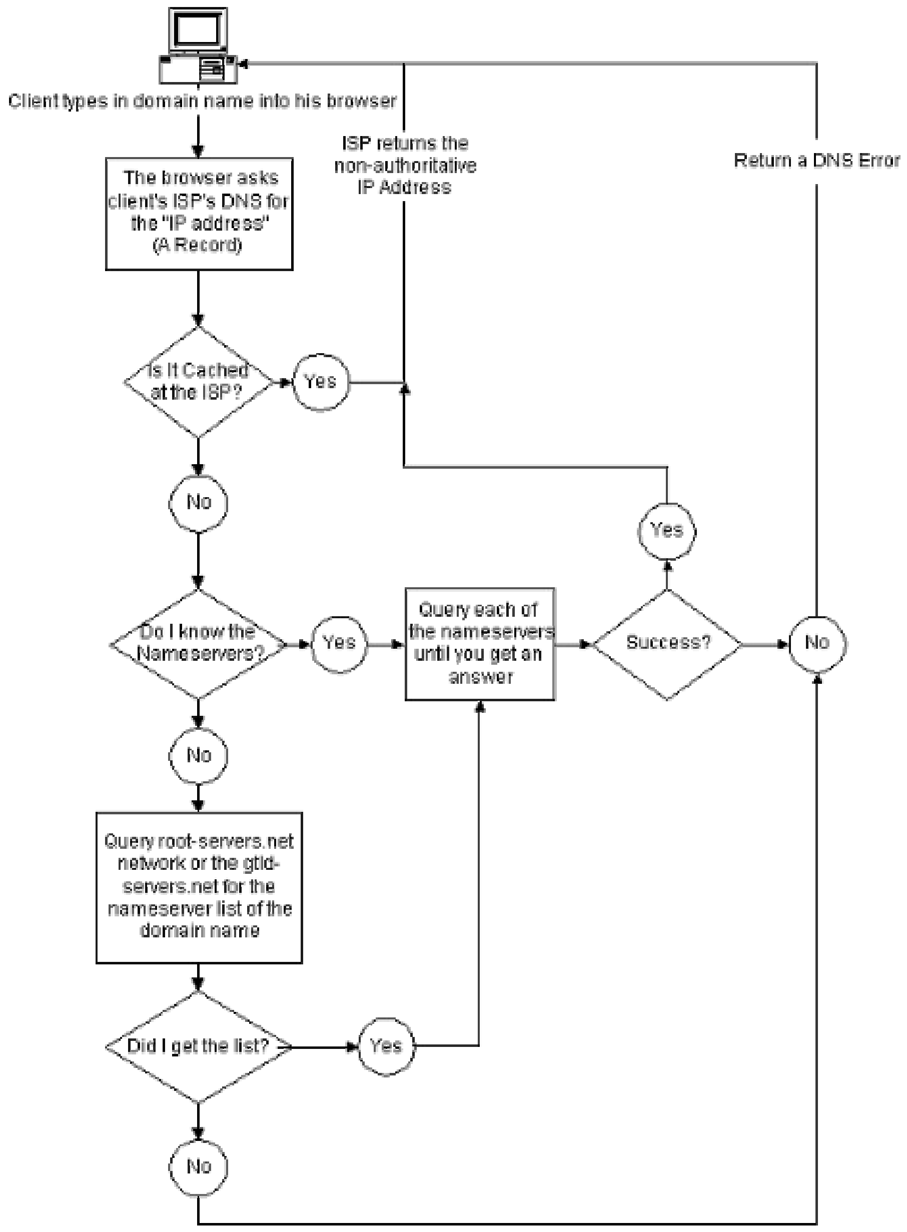

Client types in domain name into his browser
The browser asks client's ISP's DNS for the "IP address" (A Record)
ISP returns the non-authoritative IP Address
Return a DNS Error
Is It Cached at the ISP?
Yes
No
Do I know the Nameservers?
Yes
Query each of the nameservers until you get an answer
Success?
Yes
No
No
Query root-servers.net network or the gtld-servers.net for the nameserver list of the domain name
Did I get the list?
Yes
No

Figure 1: How DNS Works (ZoneEdit.Com, 2002)

Now that the users browser has the IP address of the site, it will now send a request to the site, along with the host header (the domain name) and the server will start serving the request for the page and associated images etc. to the users browser if it has been configured to provide service for the domain name or IP address the browser has asked for.

1.2.2. Web Server

It is the job of the web server software to listen for requests made for web pages on IP addresses and domains that it is configured to listen for. Once it receives a request it then finds the appropriate page/file, performs any processing on it that it needs e.g. PHP, Perl, SSI and then sends the final output to the users web browser.

As discussed in the project specification, I will be using the open source Apache from the Apache Software Foundation as the web server for the project as it provides a stable and well tested, 66% of active web sites use Apache (Netcraft 2003), platform on which to develop a suitable control system.

1.2.3. Server Hardware

All software that will be used is readily available on a number of hardware platforms such as:

Intel x86 (Intel, AMD, Cyrix)
Sun Sparc
IBM
SGI
HP

For the proof of concept I shall be using the Intel x86 architecture, as the hardware platform is stable and well tested, it is the primary platform used by the vast majority of small-medium sized companies providing shared hosting solutions. KDA Web Services Ltd. has kindly donated a server and internet connectivity for the Web and DNS servers to run from and a shared hosting account to run the control software.

Using the Intel x86 architecture means that if any problems are encountered when installing software then there is more chance of finding a solution quickly due to the fact that the x86 architecture is one of the most popular and widely used and tested for running small-medium servers and compiling software on.

1.2.4. Server Operating System

Both the control and the remote servers will run RedHat Linux 7.3 for the purpose of the proof of concept as it is a widely tested and supported distribution on the Linux operating system and is also the operating system used on the servers that CPanel and HSphere were evaluated on. Linux is an open source operating system used on a large number of web servers, mainly due to its free, open-source nature, which means anyone can use it and even alter it, if they find a bug, then they can fix it – Another benefit of the open source nature is that anyone can review the code, and lots of people do, this large scale peer review can help reduce security holes and other potential problems, thus leading to a more robust system.

1.3. Evaluation of existing solutions to the problem

As discussed earlier, I shall be looking at two existing solutions, HSphere from Positive Software Corporation and CPanel6 from cPanel Inc. both are well known and well established products, they are also both products with which I have first hand experience of installing, using, administering and supporting on my work placement year.

Criteria I shall be using for evaluation within the context of my research are:

> Customisation – How easily can the look and feel of the user interface be changed? The colours? The layout? The text, for internationalisation purposes?
>
> Expandability – Can the system easily be expanded to add new features as and when the host requires them?
>
> Scalability – Will the system grow with the company as their client base expands and can it do so in a hassle free way?
>
> Service Diversity – Can you use more than one type of web server? Per machine? Are you limited in what FTP software you can run? What databases are supported?
>
> Usability – Is the system easy for end users and administrators alike to use? Does it make their respective tasks/jobs easier? Based on a fresh default install.

1.3.1. HSphere from Positive Software Corporation

"HSphere is a scalable multiserver webhosting control panel, which provides complete hosting automation for Linux, BSD & Win2000 platforms, is easy to use, and has extensive user interface, billing solution, and integrated trouble tickets system" (Positive Software Corporation, 2003)

As the description of HSphere says, it aims to provide a single consistent interface for working with Linux, BSD and Windows 2000 hosting accounts. One of its main selling points is the way that it can place services on one server, or allocate a server a single task. Services are split into six distinct areas:

Control Server (Linux or FreeBSD)
Database Servers (Linux – mySQL or PostgreSQL, Windows - MS SQL)
DNS Servers (Bind – Linux or FreeBSD)
Mail Servers (Qmail – Linux or FreeBSD)
RealMedia Servers (Linux or Windows)
Web Servers (Apache – FreeBSD or Linux, IIS - Windows)

An example setup would be something as follows:

Server1 – Control panel
Server2 – email and DNS
Server3 – Websites
Server4 – Databases and DNS

This flexibility in where services are hosted physically can provide many benefits to a hosting company, such as allowing the physical hardware configuration and the software configuration to be tailored to the particular task it is doing. So rather than having to be a jack-of-all-trades, it can be a master of one, provided better performance and more reliable service. Working in this manner also provides a system that is highly scalable, if your web server runs out of steam, just add another one put all new sites on it, customers don’t even need to know there is anything different.

HSphere currently supports the following operating systems:

RedHat Linux 7.2 and 7.3
FreeBSD 4.3 and higher
Windows 2000 Server

Other notable features in HSphere are:

Fully integrated billing system, with support for over 15 online credit card processors
Fully integrated trouble ticket and knowledgebase system
Easy modification of control panel colour scheme

1.3.2. CPanel6 from cPanel Inc.

“The Cpanel and WebHost Manager package allows you two interfaces for web hosting control. The Cpanel interface is a client side interface, which allows your customers to easily control a web hosting account….. The WebHost Manager Interface allows Web Hosting companies to control the accounts on their servers. Through WebHost Manager you can add/remove accounts on a server, park or point domains, control bandwidth, disk space, and more.” (cPanel Inc., 2003)

At the current moment in time, Cpanel is designed with smaller hosts in mind, where they have a single server and run all services from it, with the exception of primary DNS, which can run from another server, and just recently, it has been programmed to allow mySQL services to run from a separate server. At the moment Cpanel does not provide any sort of central management for large numbers of servers, or the central management of customer data. It also does not provide any sort of inbuilt billing system or trouble ticket system.

What this means for any organization doing large scale hosting is that they will have to invest in more software for customer management or to create their own to track which server a customers website is hosted on.

Currently CPanel supports the following operating systems:

RedHat Linux: 6.2, 7.1, 7.2, 7.3 and 8.0
Mandrake Linux: 7.2, 8.0, 8.1, 8.2 and 9.0
FreeBSD: 4.2, 4.3, 4.4, 4.5, 4.6 and 4.7

Work is also underway on versions for Mac OSX, Debian Linux, Sun Solaris and Microsoft Windows.

Other notable features in CPanel are:

Support for skins, the look and feel of the admin and end user GUI can be changed by the creation of a new “skin” with each hosting plan having the ability to use a different skin.

1.3.3. How They Compare

Customisation

CPanel – CPanel has the ability to create skins, for the admin and end user interface. For the end user interface the skin used can be individual to each hosting package created.

HSphere – HSphere also has the ability to create skins for the end user interface, but it does not have as well a documented set of commands as CPanel does which makes it harder to create you own skins. Again, these skins can be used on a per package or even per user basis. HSphere also has the added ability to control the colours in the default skin from an easy to use control panel – See Appendix B, Fig 3.

Expandability

CPanel – The only way custom features can be added is to create or modify a theme to include them in it.

HSphere – The only way custom features can be added it to create or modify a theme to include them in it, although an API to allow custom features to be added from the hosting package creation wizard is in development.

Scalability

CPanel – Has no built in support for central management of customer records, all records for customers are stored on the server their website is on, and duplicated if they have more than one account. All data about accounts is stored in text files – not providing a very robust and scalable solution.

HSphere – Manages all customer and account data from a central server, no duplication of data as customers can have multiple accounts under one username. HSphere uses a database to store all details, by default PostgreSQL, but Oracle or MS SQL can also be used.

Service Diversity

CPanel – Only currently works on Linux and BSD, but does offer a choice of different FTP software to use. Work is currently under way to support other operating systems as well. It currently supports more Linux distributions than HSphere.

HSphere – Currently works with FreeBSD, Linux and Windows, and offers a fairly good range of services.

Usability

CPanel – In the default fresh install the end user and admin interfaces are well organised and easy to find items as the sections are clearly labelled and items are not hidden in areas you would not expect to find them. Generally users find the default interface to CPanel and WHM much more user friendly and usable than they do for HSphere.

HSphere – Admin and end user interfaces are a little disorganised and it can be hard to find some items as they are under menu sections that you would not reasonably expect them to be under. This can lead to confusion on the part of users and increased support issues raised due to users not being able to find items to make changes themselves.

Overall

	CPanel	**HSphere**
Customisation	5	6
Expandability	3	3
Scalability	4	10
Service Diversity	5	7
Usability	7	5
TOTAL	**24/50**	**31/50**

1.4. Drawbacks of the existing approaches

From looking back at the scores for both HSphere and CPanel in the previous section we can see that neither CPanel nor HSphere is an ideal solution for the areas identified as being important. Whilst HSphere comes closest to the ideals, and perhaps with future developments mentioned, it will score higher, something more is needed for large scale hosting organisations.

1.4.1. CPanel

CPanel falls down on several areas, one of the most fundamental being scalability, it was designed from the start to manage single web servers not large clusters of servers for virtual hosting. What this means at the moment is that CPanel has no central repository for user data, such as name and address information, as well as for hosting account data – this can make every day tasks such as billing quite difficult. Another point of note on scalability is that CPanel stores all data in flat text files, which can be very inefficient with large amounts of data or for relational data – unless data is duplicated in several locations, which is the case with CPanel.

CPanel also does not integrate any notion of customer billing in to it, in any medium to large sized organisation this can pose a large problem as billing can take up a substantial amount of time, especially when you have to check if customers have used over their allotted limits for items such as disk space and data transfer – with CPanel in its default state, it can take roughly 10-15 minutes per account, as firstly you have to find the server the customer has an account on, then find the account on the server to see how much disk space it is using, then find it in another screen to see how much data transfer it has used, then actually bill the customer.

The final major failing against the criteria is that CPanel is not easily expandable, there are currently no ways in which to fully integrate your own add-ons for it e.g. Add an option for it to the package creation menu, so that your feature can be enabled on a per package basis. This combined with the fairly limited set of software which can be used with CPanel, does not provide a hosting company a way to offer a diverse range of services under a single common interface.

A major plus point for CPanel is the documented API for use when creating new skins for the end user and admin GUI, what this means for an organisation is that they can create a distinct branded look and feel to the user experience, rather than having an interface which is instantly recognisable as that of CPanel. This allows a hosting company to distinguish themselves a little from their competitors, which is never a bad thing.

1.4.2. HSphere

HSphere, the same as CPanel falls down on a couple of areas, the main one being that of expandability. At the moment there is no documented API available for anyone who wishes to add their own modules in to HSphere, which makes it very difficult to add new features or extend current ones.

HSphere, unlike CPanel does not suffer so much on the diversity of services, as it provides the choice of 3 database systems (mySQL, PostgreSQL and MS SQL), support for Real Media streaming on both Linux and Windows servers as well as Windows 2000 based hosting with ASP support. This diversity gives hosts the ability to provide a wide range of offerings all managed from a central location and providing a consistent interface to the customer.

On the usability side, HSphere is currently lacking, as it stands at the moment with the default skin, the user interface can be inconsistent and confusing, one such example is with monetary values, in some places you will see:

£7.95

In others:

7.95 £

This, mixed in with inconsistent use of () in billing, usually used to indicate a negative value in accounting, and combined with the fact that some commands are not in logical places can cause confusion to users and thus increase the support workload of a host.

1.5. Tools for implementation

With the abundance of programming languages available it can often be difficult to pick the right language to carry out a particular task in, to make the decision easier it is worth ignoring the language specifics for a while and looking at the execution method of the resultant programs. The execution method can be categorised in to 3 different types for most modern day languages:

- Native compilers
  - Compile source code in to a binary suitable for a particular hardware and operating system (OS) combination.
- Just-In-Time (JIT) compilers
  - Compile source code in to a platform independent intermediate format, sometimes called byte-code, the byte-code can then be run on any platform that has a virtual machine written for it to execute the byte-code.
- Parsers
  - Parsers do not compile code in to an executable, they parse the code every time it is run and convert it into machine executable code, run it, then dump it – although some systems will cache the compiled code in memory for when it is needed again, but this mainly relates to web servers (Zend Accelerator for PHP and mod_perl for Perl).

Each execution method has it's advantages and disadvantages which I shall now cover:

	Advantages	**Disadvantages**
Native Compiler	Faster program execution	Needs to be compiled for

	due to native compilation for the architecture.	every machine architecture the program will run on.
Just-In-Time	Compile once, run anywhere, allows for high degree of portability.	Can be easy to reverse engineer the byte-code to the original (or near enough) source code.
Parser	No need for compilation after a code change, allows for rapid prototyping. No need for compiling, so will run anywhere that the interpreter does.	Slow execution compared to compiled programs as code has to be interpreted every time it is run.

Examples of natively compiled languages are:

- C/C++
- Assembly Code

Examples of Just-In-Time compiled languages are:

- Java
- Microsoft .NET

Examples of parsed languages, sometimes known as scripting languages are:

- PHP
- Perl

- Python

For the development of the proof of concept application, I shall be considering 3 languages, C/C++, Java and PHP and have compared them below on key criteria for any development project.

	C/C++	**Java**	**PHP**
Learning Curve	3	5	7
Ease of web use	3	9	10
Execution Performance	9	7	6
Development Speed	5	7	7
Development Ease	4	6	8
Scalability	7	8	4
TOTAL	**31**	**42**	**42**

From the above comparison we can see that both Java and PHP figure strongly for use in the development of the system, as it is only going to be a proof of concept and due to the time constraints and my own overwhelming experience with PHP I believe it would be the best choice for the proof. PHP will allow the rapid testing of any changes to the code base as it does not need to be compiled before running.

In a production system it would be wiser to use Java as it provides features such as object persistence so that objects are still in memory after an operation has been executed, unlike PHP where objects die once the requested piece of code has finished executing. Java also provides advanced support for databases, such as connection pooling which can reduce the load placed on a database server and help conserve system resources and above all, Java on the server side has been designed with scalable enterprise applications in mind, whereas PHP was conceived to save time on developing personal home pages.

1.6. My approach

From my research in to existing solutions, I believe the best approach to take is one similar to what HSphere takes, to have a central control server which is in charge of storing all configuration data and customer data in a central repository. As well as storing the data I believe the server should be in charge of processing the data in to the relevant format for the server, so that it may be written straight to the configuration file for it - without the client machine having to do any further processing. What this means is that any issues with the control server can be debugged on the control server and not have to involve changing code across multiple servers to debug/fix the problem, thus simplifying the development and ongoing maintenance process.

2. System Design

2.1. Design notations

2.1.1. UML

“The Unified Modeling Language™ (UML) is the industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. It simplifies the complex process of software design, making a "blueprint" for construction.” (Rational Software, 2003)

UML uses 9 types of diagrams to document the design of a system, they are:

- Class Diagrams
  - Describe the structure of the system, what classes/object it will use, the methods and data members they have and how they relate to each other.

- Package Diagrams
  - Used to organise related elements of a system in to groups, to help reduce the number of dependencies between them.

- Object Diagrams
  - Object diagrams are used to describe the structure of a system at a particular point in time and can be thought of as an instance of a class diagram at a particular point in time.

- Use Case Diagrams
  - A use case diagram is used to model the functionality of a system in terms of use cases and actors (users). A use case is a

method/function provided by the system e.g. Print Document, Spell Check.

- Sequence Diagrams
  - A sequence diagram shows a particular execution path through a system, from the point of user interaction e.g. What happens in terms of classes and method calls/messages when a user clicks the print button in an application.

- Collaboration Diagrams
  - A collaboration diagram is used to describe the interactions (In terms of sequenced messages) in a system between objects.

- State Diagrams
  - A state diagram shows what happens when part of the system changes from one state to another e.g. A button is clicked by a user.

- Activity Diagrams
  - An activity is an operation on an object that results in a change of state. It is the job of an activity diagram to show the flow of control between activities.

- Component Diagrams
  - A component diagram is used to show the organisation of the physical software components in the system e.g. The source code of the system, the binary executable files, the resources etc.

- Deployment Diagrams

  - Deployment diagrams show how the physical resources of a system are organised e.g. Server nodes, client nodes and what components reside in/on each physical resource.

UML is not a methodology as such, more a general systems development notation, hence UML is ideal for documenting the design of a wide range of systems, from small proofs, all the way to large scale enterprise systems due to the large variety of pre-defined constructs within it and the fact that it is not rigidly set in stone – If you don't want to produce one type of diagram because it is not needed, then you don't have too for it to work effectively for you.

2.1.2. SSADM

Structured Systems Analysis and Design Methodology (SSADM) is a widely used development method often specified as a requirement for UK government computing projects and is specified in BS7738. It aims to improve the control and project management aspects of development, make better use of both inexperienced and experienced staff whilst making projects resilient against the loss of any member(s) of staff and to above all, produce better quality systems as a result of applying it.

SSADM is comprised of 6 stages, which are:

- Stage 1 – Investigate the current environment
  - Draw a Data Flow Diagram (DFD) and a Logical Data Model (LDM) showing the current system.

- Stage 2 – Business Systems Options (BSOs)
  - Describe possible new systems in terms of functionality and implementation issues. Use text and skeletal DFDs and LDMs.

- Stage 3 – Requirements Specification
    - After choosing a BSO, refine DFDs and LDMs. To model how the system will respond to events (entity behaviour modelling), draw and entity life history (ELH) diagram, an effect correspondence diagram (ECD) and enquiry access paths (EAP).

- Stage 4 – Technical System Options (TSOs)
    - Describe the costs, benefits and constraints if implementing the specification.

- Stage 5 – Logical Design
    - Define how data is processed by the system and describe user dialogs. Update the entity life history diagrams with state indicators and draw and updated processing model (UPM)

- Stage 6 – Physical Design
    - Develop user interface structures and implement logical processes.

(Smartdraw, 2003)

SSADM is a structured methodology, which was originally conceived for use when developing information systems, hence is tailored towards systems that place a great deal of emphasis on the information they store e.g. a bookings system for a hotel. SSADM also follows the waterfall approach, meaning that you have to start at the top of SSADM at stage one and work your way through all 6 stages, completing and signing off each stage as it is finished – This means it is very rigid and does not offer a great deal of flexibility.

2.1.3. Conclusions

I feel that UML is the more appropriate tool for documenting the design of the system in discussion, as we are only dealing with a proof of concept deliverable based on the ideas laid out. I believe that using SSADM would take up too much time from other areas of the project, whilst it provides a very structured approach to the whole development life-cycle it can be easy to get bogged down and stop making any real progress, which considering the time constraints set out, would not be wise to let happen, therefore UML shall be my choice for documenting the system.

2.2. Design and development methodologies

Throughout this section I will use a system as an example where information about Users and Visitors is stored, a User being a member of staff who can login to a companies system and a Visitor being someone visiting the company.

When designing systems there are two approaches we can take to the development of the code for the deliverable, we can take the procedural approach or the object oriented (OO) approach to developing our code.

2.2.1. Procedural design

Code is usually organised in to functions, and to make organisation easier it is nearly always organised in to appropriately named files too e.g. User.c, Guest.c, Visitor.c although there is no requirement to do so, neither is there a requirement to place code in to functions in most languages. So in the worst case with procedural programming we could end up with a file looking like:

```
getUserName()
getVisitorCompany()
getUserLastLogin()
getVisitorName()
getUserLoginName()
getVisitorLastVisit()
```

As you can see, it’s not very ordered and organised, even for a small application it could get very messy, very quickly and be difficult to track down where various functions reside and how they relate to each other. Also a problem is data encapsulation, if you have global variables that related functions need to have access too, you have no way of stopping unrelated functions interfering with them – this can cause many problems and be very difficult to debug, for example:

You have a variable, counter, that you rely on to keep track of how many times a certain type of operation has been performed, e.g. a query to a certain database table. Now if another set of functions also wanted to use a counter to keep track of an operation, they would have to make sure that they used a different global counter variable. If they did not, then they could end up incrementing the same counter, causing problems with the counter being incremented faster than the functions that used it expect it to be, possibly causing problems if they rely on the value for any calculations or for triggering events. This can be hard to debug as you would have to find all functions that use the global counter variable and change them to use their own variable, but you would also have to find all related functions and change them to use the same related counter variable, as you can see, it can get quite complex.

2.2.2. Object Oriented design

Code functions are organized into logical objects e.g. all methods and data elements for working with user data (Information about a person) could be stored in a User object with methods:

```
getName()
getLoginName()
getLastLogin()
```

And a Visitor object (For visitors to our example company):

```
getName()
getLastVisit()
getCompany()
```

As you can see, we have a method in common between the Visitor and User objects, getName(). To take our design a step further we can utilise one of the fundamentals of OO design, sub-classing or extending objects. By taking out common elements for a Person (Visitor and User are both persons at a base level) we create a Person object with the following methods:

```
getName()
getAge()
```

Now we change our User object, to extend the Person object with the following methods:

```
getLoginName()
getLastLogin()
```

We also change our Visitor object to extend the Person object and add the following methods:

```
getLastVisit()
getCompanyName()
```

Moving out the common features of User and Visitor means that in the future, should we need to change the way any of the common methods work e.g. getName() We only have to change it in the Person object and the User and Visitor objects won't notice any difference and won't have to be changed. If we had left getName() in both User and Visitor then we would have to change both of the objects to reflect the new way of coding getName(). Doing so is not ideal as it can lead to inconsistencies in code that can be very difficult to track down, especially once a system is in production use.

Other advantages of OO design and construction are:

Encapsulation – We can define what other functions can have access to variables in the object e.g. Only methods in the Person object can change the age data member, but the Person object and all sub-classes can change the name data member. Or we can make data members available so that any method in any object can access them, something which is not advisable though.

Object Oriented design and construction is not always the best approach for all software as it can add extra programming overhead and in interpreted languages in can also add extra processing time in some cases. OO design is really only suited to developments where you have discreet data objects where the data and methods for manipulating the data can be contained in a single object, or where there are many variations on a theme e.g. the Person, User, Visitor example or in the case of this system, Server, Web Server, Apache Web Server.

2.2.3. Conclusions

It is my belief that OO design principals and practices are best suited to this project over procedural design, due to the fact that it is mainly comprised of discreet entities that require data storing about them and/or working on e.g. User, Domain Name, Server etc with some of them being of a similar theme and suitable for sub-classing e.g. Apache Web Server is a sub-class of Web Server which is a sub-class of Server.

2.2.4. Design Patterns

"Each pattern is a three-part rule which expresses a relation between a certain context, a problem & a solution" (Dearden, 2003)

Taking OO design a step further, we come in to the realm of design patterns, each pattern is a solution to a common design problem, which can be used when solving your own design problems. Design patterns are not just specific to software engineering, they started out in the realm of engineering with a man named Christopher Alexander coining the phrase "Pattern Language" as the way we describe abstract solutions to recurring problems.

One such pattern is the abstract factory pattern, it describes how we can allow the creation of one of a family of related objects, leaving the decision on which object to create till the time of object instantiation, without having to hard code if/switch statements to make the decision everywhere we need to instantiate an object from the family.

For example, in the design of our system, we will need to control the web and DNS servers, of which we can have multiple types e.g. Apache Web Server or Zeus Web server and Bind DNS Server or DJBDNS Server. To solve this, we would create an abstract WebServer factory and an abstract DNSServer factory, which would handle the specifics of which web/DNS server to create based on a variable set at runtime in the system. The client application would ask the WebServer factory for the web server object, passing it the variable for what type it would like, the WebServer factory would then instantiate a sub-class of the WebServer class that implemented the correct functionality for the server requested. The application would then use the returned object without ever knowing that it was a different object to what it requested, as the object is a sub-class of the abstract WebServer, implementing all the required methods for operation – Thus, the implementation specifics for each type of web server are hidden from the client.

2.3. The Design

Taking in to account the above design decisions and the requirement of the system to be easily expandable/module it seems prudent to make use of the Abstract Factory Pattern (AFP) in the design of the system. The AFP tells us to define abstract classes for a factory that define the methods a concrete factory should provide, how the concrete factory implemented the methods does not matter to the application using the factory or the products of the factory. We also need to define abstract and concrete products when applying the AFP, again, the abstract product defines the methods a concrete product should implement, but not how it should. Using the abstract + concrete approach allows us to provide a consistent object interface to any application utilising the factory/product whilst hiding the implementation and ultimate class type of the factory/product objects from the application – What this means is that we can easily add new factories/products based on the abstract factory/product without having to make any changes to the client application if we so wish, ideal for our application.

With this in mind, it would make sense to create a Concrete Factory for each type of service we want to support e.g. Web, STMP, POP, DNS, FTP etc. Or in the case of our proof of concept, just Web and DNS, then we would create Concrete Products for each Factory which represent the various applications that can be used to host the particular service, so for the Web Factory we create an Apache concrete product and for the DNS Factory we create DJBDNS and Bind Concrete Products. From this, we can produce a highly simplified class diagram showing how the various Abstract and Concrete classes relate to each other, as can be seen in Figure 2 below.

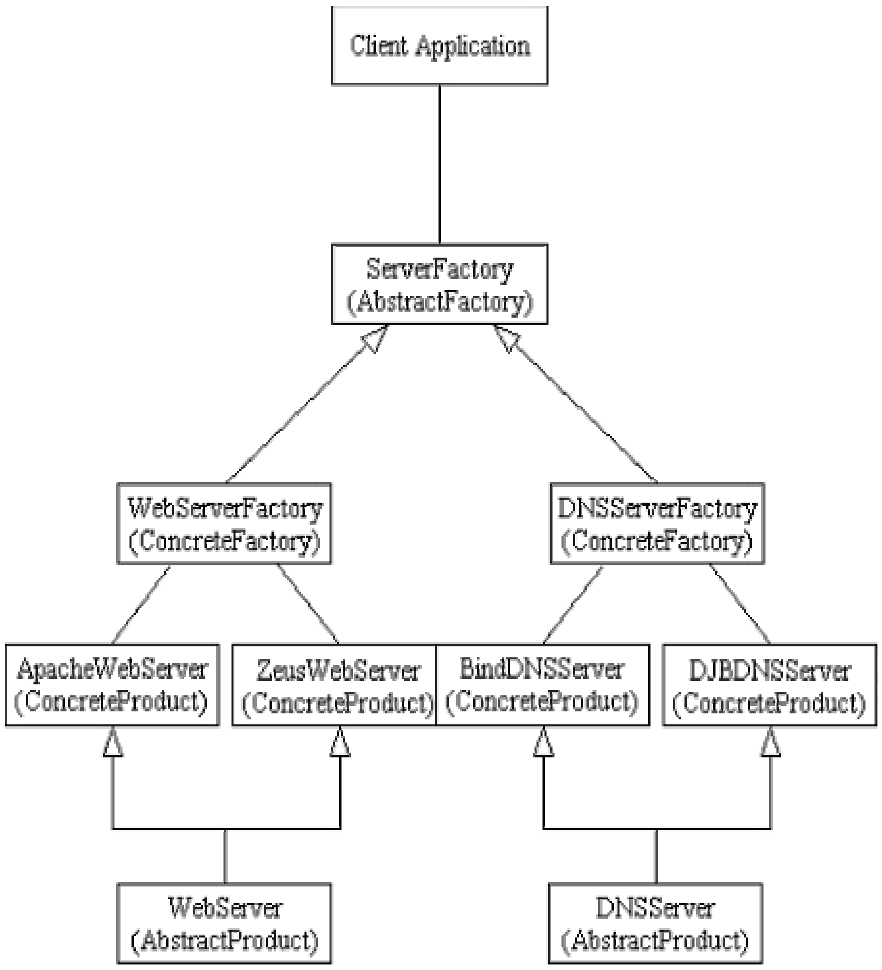


Figure 2: Applying the Factory Pattern

If we take the above classes, we can produce a sequence diagram of what would happen when creating a new configuration for the Apache web server, as seen in Figure 3 below.

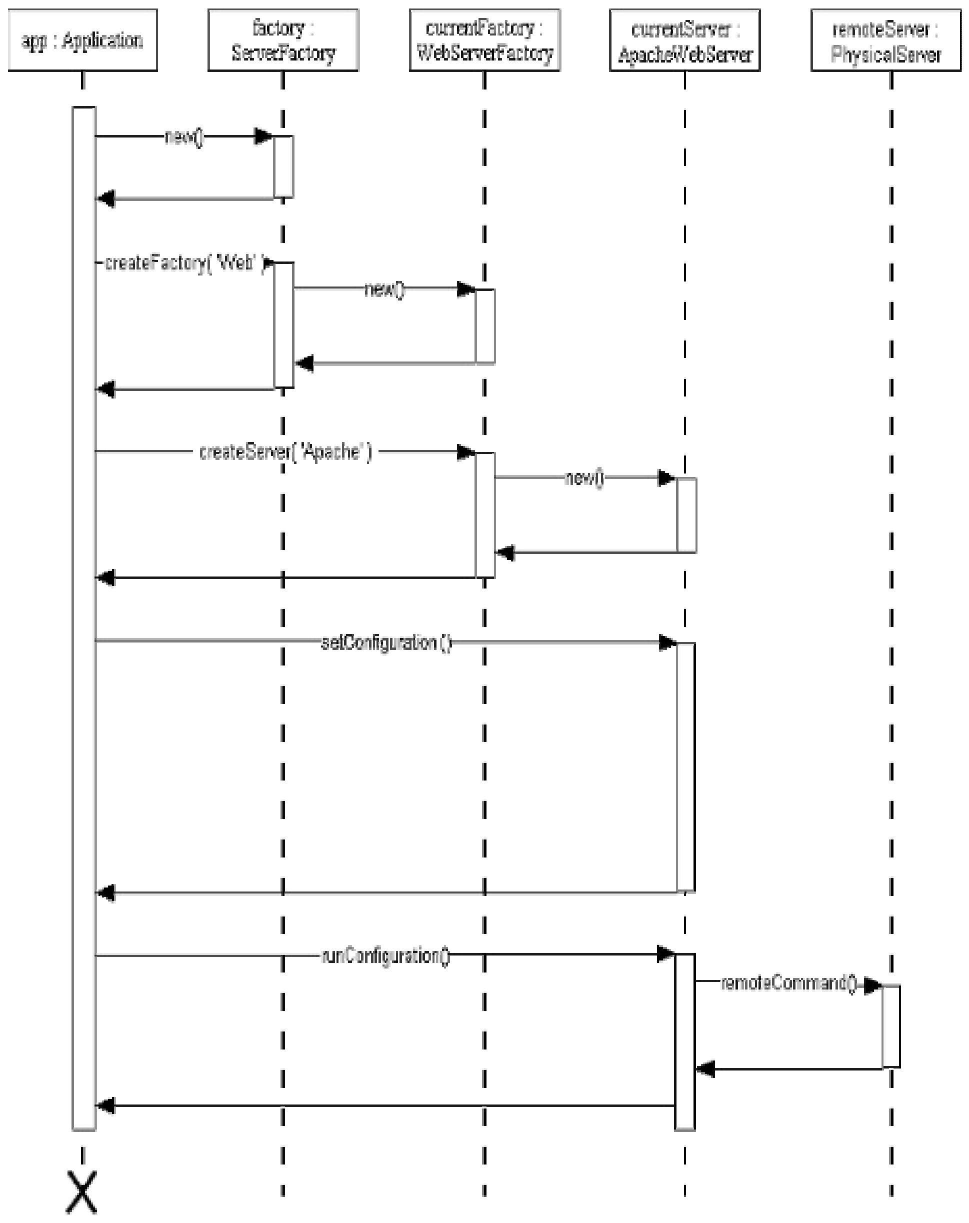


Figure 3: Sequence diagram for creating a new Apache web site.

As you can see, the first task is to ask the main factory class for a WebServer factory which can then be used to fetch an instance of a WebServer sub-class, in this case ApacheWebServer. After this happens, the application sets the configuration of ApacheWebServer, then it instructs the class to run the configuration – which as some stage involves connecting to a remote server to execute commands on it, to configure the service.

A similar set of events would also happen for configuring the Bind of DJBDNS servers, except a DNSServer factory would be asked for instead of a WebServer factory and currentServer would be of the appropriate concrete product class.

As the program deliverable is only as a proof on ideas, I believe this is enough of a system design to work from, it is only minimal, but the main focus of the project is to look in to ideas for modularity in the coding of the system, and this cannot be shown all that well through design without having many very similar diagrams which serve no purpose when implementing the system – Thus I feel it is better to let the development and the implementation actually do the work of showing off the ideas.

3. System Development

3.1. Implementation considerations

One item of particular importance when developing a distributed system is the approach taken to distributing the workload between the client and the server. Two common approaches are:

Thin client, fat server – The server does the vast majority of the work, then instructs the client what to do.

Thin server, fat client – The server just sends any data to the client, the client then handles any processing of the data.

Each system has it's advantages and disadvantages, but before looking at them, it is worth outlining how each of the two approaches would apply to the system in discussion.

Thin client – The server would process any data, in to a format that the client can understand without the need for any additional software to be installed on the server other than what comes with the operating system and also without the need for any additional processing of the information passed from the server. For example, when setting up a new web site, the server would take all the configuration information, turn it into a block of commands that the client would understand e.g. echo this data into this file. The server would then pass the command block to the client for running.

Thin server – The server would take any data and pass it to the correct client for processing, the client would then handle the tasks of working out what to do with the data. For example, when setting up a new web site, the server would pass all the data needed for configuration to the client, the client would then process the data into configuration commands for the software and execute them.

	Advantages	**Disadvantages**
Thin Client	Keeps all processing code in one location, so that if it is changed, it only needs updating on the server, not on multiple client machines. Less overhead placed on the client server, meaning more CPU time can be spent on doing it's actual job rather than processing configuration data.	Places extra load on the server as it has to carry out more of the processing tasks.
Thin Server	Removes a lot of the processing from the server, meaning a less powerful server can be used or more clients can be serviced from each server.	Takes the client machine away from just carrying out its specific task, it now has to deal with processing data into configurations.

3.2. Communication protocols

One of the most important aspects when designing any distributed system is the communication medium that will be used and the security of it. The security of the communication channel becomes even more important when the Internet is involved as it is an open and public network, where any node on the network has the ability to listen in on traffic passing through it.

There are two main protocols for securing communications over the Internet, these are:

Secure Socket Layer/Transport Layer Security (SSL/TLS)

and

Secure Shell (SSH)

SSL/TLS is the dominant protocol for securing data transmission over HTTP as it is used for secure server certificates to provide https functionality to web servers, it can mainly be seen securing the numerous e-commerce sites on the Internet. SSL isn't however used exclusively for securing HTTP traffic, it can be used for securing any sort of traffic over TCP/IP and is used to secure FTP, SMTP and POP3 protocols too, as well as many custom built applications.

SSH is the dominant protocol for remote console connections over IP to servers, it is a secure replacement for RSH (Remote Shell) which provides a text based console, just as if you were sat at the server. SSH can also be used to tunnel other protocols through it, such as FTP, providing a more secure version of FTP as the communication channel to the server is passed over SSH thus encrypting the communication of commands, usernames and passwords.

For this particular application, a combination of SSL/TLS and SSH would be the best approach, with SSL/TLS being used to secure the web interface, so that passwords and user details cannot be packet sniffed for. SSH would then be used between the control server and the remote servers for executing the commands to configure services as it has options for logging in to servers without a password – as long as the server trying to login to the remote server has their public key installed on the remote server, this feature makes it ideal for use in remote communications between servers, especially when coupled with the ability to automatically run a command on connection when using the OpenSSH client e.g.

```
    ssh root@server.example.com “ls”
```

The above command would execute the Unix ls (Similar to the DOS dir command) command once it has connected as the root user on the server server.example.com. This feature makes it very easy to use the SSH client from within any program, as you do not have to pipe the command you want to execute into a running process of the SSH client – which can be troublesome on some operating systems, especially if they do not support reading and writing from the same pipe, only one or the other.

3.3. Data storage

3.3.1. Configuration data

For configuration data, I believe it would be best to have it stored in text flat text files, as it makes it very simple to change any items with just a text editor over a secure shell session e.g. from a PDA or Mobile Phone. Reading data from simple text files is also considerably quicker than connecting to a database and running a query to get the data, as long as there are no complex rules to reading the configuration file – else a database will be faster.

For each concrete product, there will be a set amount of information stored, which will include:

Commands to start, stop, restart and reload the application the concrete product controls
The commands needed to configure a new item/account/user on the product

I propose to store this data in an XML file, with the following structure:

```
<module name="" type="">
      <commands>
            <start></start>
            <stop></stop>
            <reload></reload>
            <restart></restart>
      </commands>
      <configuration type="">
            <file order="" type="" test="">
                  <location></location>
                  <data></data>
            </file>
      </configuration>
</module>
```

The top level module element, has attributes for the name of the module and its type, in the case of Apache these would be, name=”Apache”, type=”Web”

The commands block is pretty self explanatory, it stores the commands needed for starting, stopping, reloading and restarting the application that the module controls e.g. the Apache service on a server.

The configuration block is where the real work is done in configuring the application, the type attribute is used to store what configuration method the application uses, at the moment it can only have the value “file”, to indicate that the application uses textual configuration files. In theory it can be adapted to support applications that use a database, or a special application on the server to handle the configuration of the application.

The file element is used to indicate a single file used in the configuration of the application, it has an order attribute which is used to specify in which order the files are to be written if there is more than one, the type attribute is used to indicate if the file in question should be overwritten (truncate) or added too (append). The test attribute is used to store a data value that will be checked for in a file when appending to it, if the value exists in the file, then it will not be added again to the file. The location child element stores the physical location of the file on the remote server and the data child element stores the actual data that should be written to the file.

If we put all this together, we have a complete XML configuration descriptor for the Apache web server:

```
<module name="Apache" type="Web">
      <commands>
            <start>/meta/apps/apache/bin/apachectl start</start>
            <stop>/meta/apps/apache/bin/apachectl stop</stop>
            <reload>/meta/apps/apache/bin/apachectl reload</reload>
            <restart>/meta/apps/apache/bin/apachectl restart</restart>
      </commands>
      <configuration type="file">
            <file order="1" type="append" test="**domainid.conf">
                  <location>/meta/config/apache/vhosts.conf</location>
                  <data>Include 
/meta/config/apache/**domainid.conf</data>
            </file>
            <file order="2" type="truncate">

      <location>/meta/config/apache/**domainid.conf</location>
                  <data>
                        <![CDATA[
                              **configoutput
                        ]]>
                  </data>
            </file>
      </configuration>
</module>
```

The ** that you can see in the configuration descriptor above are used to denote the start of a variable that will be replaced later on in the execution of the application e.g. **configoutput will be replaced with the actual data to be written to the configuration file.

Using the above configuration descriptor has allowed me to develop a generic configuration routine that can configure any of the current concrete products and any future concrete products that use the file based configuration method. What it also means is that there is no complex programming logic to decide which concrete product we are configuring and how to actually go about configuring it.

3.3.2. General system data

General system data for this application can consist of:

User data
Personal Details
Details of any accounts they have
Account Data
Configuration settings for domains e.g. DNS, Apache
Details on hosting accounts, such as number of email address allowed etc.

Due to the large quantities of data that can be generated by the above and the sometimes complex relationships between the data, it is better suited to storage in a relational database system, such as PostgreSQL, Oracle, Sybase ASE etc. however due to time constraints imposed on the development of the proof of concept it has not been possible to make use of a relational database system and only details on the DNS and Web server settings for a test domain have been stored, in an XML formatted text file, as shown in Appendix C.

The data for the Web Server includes:

Domain Id, Domain name, Domain IP, Domain Aliases
Location of web files, location and format of log files
Location of custom error pages
Configuration of and file extensions for, CGI, PHP, SSI
Options for directory indexing

The data for the DNS Server includes:

Domain name, Domain IP
DNS record Serial number, refresh time, retry time, expiry time, time-to-live, start of authority and the hostmaster email address
Details of sub-domains, mail servers, and cname aliases
Options for controlling reverse IP lookups and domain wildcarding (enabling *.example.com to function) are also included

3.4. Turning data in to configurations

For the system to actually be useful, we have to turn data about the domains and settings in to actual usable configurations, this may mean taking the data and formatting it into a specific configuration syntax or formatting it in to SQL queries to be executed. The most common approach to this would be to create a block of program code for each application we wanted to control, and use the code to output the configuration commands interspersed with the configuration data stored in the system. Whilst this approach works and is simple, it is not an ideal situation and does not lend itself well to future expandability.

3.4.1. Enter XSLT

The World Wide Web Consortium (W3C) defines XSLT as:

"a language for transforming XML documents into other XML documents"

So how can it be useful in the system? The W3C defines XSLT as transforming XML into another XML format, however this does not have to be the case with most XSLT engines on the market, most can also transform XML in to plain text, which is where it becomes useful to us. We can use XSLT to transform our XML configuration data in to any format that we please now, meaning we can take our Apache configuration descriptor and turn it in to some PHP code that can be executed as shown below.

```
$command['start'] = '/meta/apps/apache/bin/apachectl start';
$command['stop'] = '/meta/apps/apache/bin/apachectl stop';
$command['reload'] = '/meta/apps/apache/bin/apachectl reload';
$command['restart'] = '/meta/apps/apache/bin/apachectl restart';


$file[1]['io'] = 'append';
$file[1]['location'] = '/meta/config/apache/vhosts.conf';
$file[1]['data'] = 'Include /meta/config/apache/10001.conf';
$file[1]['test'] = '10001.conf';
```

```
$file[2]['io'] = 'truncate';
$file[2]['location'] = '/meta/config/apache/10001.conf';
$file[2]['data'] = '

<VirtualHost 62.149.37.18>

<<snip rest of apache configuration>>

</VirtualHost>

';
$file[2]['test'] = '';
```

We aren't just limited to converting to PHP, we can convert to any format we can create an XSLT style sheet for – which is pretty much any ASCII format conceivable. The use of XSLT to transform our data allows us great flexibility and power within the system, for example, to create the admin GUI to control the configuration descriptor, all that would be needed now is an XSLT style sheet to convert the XML into an HTML form to be displayed in a web browser, and to create the code to save the data when submitted back in to XML format – The style sheet would then be used for the editing of all configuration descriptors, thus providing a consistent user interface, whose look and feel can be controlled from one file whilst effecting the look and feel of all configuration descriptor editors. The same scenario can be applied to the actual data stored for the web and DNS server configuration, except we could provide and administrator XSLT style sheet, that allows all of the items to be edited and saved and an end user XSLT style sheet, that only allows them to change certain aspects of the data e.g. Prevent them from changing the home directory of their domain name, which would prevent them setting it to that of another customer.

3.5. Verification and validation

As the program deliverable is only relatively small in terms of features used, it is quite easy to validate whether it is functioning correctly. As its main role is to output configuration data, we can check the validity of this at any time by running the syntax checker for the various applications.

For Apache configurations:

```
apachectl configtest
```

This will open up the Apache configuration file and check it for any syntax errors or other problems that would stop Apache from running.

For BIND zone files:

```
named-checkzone karlaustin.com karlaustin.com.bind.conf
```

This will check the DNS zone described in karlaustin.com.bind.conf and see if it produces all the correct items for the domain karlaustin.com such as SOA record and other mandatory items, as well as checking that other configuration lines are of the correct format to be used.

To check for correct formatting of DJBDNS configuration files:

```
tinydns-data
```

This will recompile the DJBDNS data file in to the binary format that it understands, if there is a problem then it will print an error message to that effect.

We can also verify that the configurations are correct by actually trying them, in the case of Apache this involves visiting a web site configured by the system and verifying that the pages within it are served correctly. For DNS we can do a query against the DNS server to see what information it has for a domain that the system has configured:

```
[root@rogue ip]# nslookup -sil
> set type=any
> server rogue1.kdawebservices.com
Default server: rogue1.kdawebservices.com
Address: 62.149.37.63#53
> karlaustin.com
Server:         rogue1.kdawebservices.com
Address:        62.149.37.63#53

Name:   karlaustin.com
Address: 62.149.37.15
karlaustin.com
        origin = rogue1.kdawebservices.com
        mail addr = hostmaster.karlaustin.net
        serial = 2002111317
        refresh = 10800
        retry = 3600
        expire = 604800
        minimum = 86400
karlaustin.com  nameserver = rogue2.kdawebservices.com.
karlaustin.com  nameserver = rogue1.kdawebservices.com.
karlaustin.com  mail exchanger = 10 mail.karlaustin.com.
karlaustin.com  mail exchanger = 20 mail2.karlaustin.com.
```

We can now compare the above output, to what we would have expected from looking at our XML data file with information for the domain in it. If we do this and compare it to the DNS data XML file in Appendix C, we can see that the domain was indeed supposed to resolve to IP address 62.149.37.15 and that there should have been two MX (mail exchanger) records, thus verifying that the DNS zone was created correctly.

4. The deliverable

The main deliverable of the project will be a set of ideas and solutions for developing modular software and a proof of concept application to demonstrate the ideas, the application will demonstrate:

- How we can code a plug-in type interface to a system, so that it does not need to know about all possible modules it will execute
    - This is a key issue to solving the problems of system expandability as outlined in the project specification. If a generic interface in to the system can be created then the control software should be able to work with and control any current and future software we may wish to use, without having to perform any form of extensive upgrading.

- How we can communicate between two servers securely from inside a program and execute applications on the remote server.
    - This is a central issue to any hosting automation system that relies on a client-server model, if communication is not secure, then it could lead to the whole system being compromised and result in a highly embarrassing situation for a host.

- How we can make significant parts of the program implementation language independent
    - If we can make parts of the program language independent, then we move a step closer to being platform independent as was one of the goals of the system. If we remove the dependency on the implementation language then we remove the need for their to be a compiler/virtual machine/parser for the machine architecture to be available.

The application will be able to:

- Create configurations for Apache web server
- Create configurations for BIND and DJBDNS DNS servers
    - Doing this will demonstrate how the system can create two separate application configurations from a common data and how the modularity is built in to the system.

- Control the starting, stopping, reloading and restarting of Apache, BIND and DJBDNS
    - Doing this will show how applications on one server may be securely controlled by another server in a remote location.

As the application is only a proof of concept on a series of ideas, it is important to not get too carried away with it, as it could turn in to a very large body of work if we started to replicate the facilities and functions offered by CPanel and HSphere – As these projects have been in development for several years now and not for the 300 hours allocated for this whole project which includes research and write-up, not just development.

5. Critical evaluation

5.1. The process

After going through the whole process from initial idea to deliverable and firm set of ideas on developing modularity in application design and coding it is clear to me that more time should have been spent on creating an action plan for the development of the application deliverable and for scheduling the many tasks that needed to be undertaken.

Whilst the design of the software is minimal, this is due to the fact that the software deliverable was only ever intended to be a proof of concept of the modular software development ideas – Therefore it was not appropriate to spend a substantial amount of time on the design on a throw-away proof of concept, in my opinion the time saved on design was much better spent gaining a better understanding of how HSphere and CPanel function.

5.2. The deliverable

The deliverable has met the majority of its goals, with the exception of having a nice graphical user interface (GUI) to demonstrate it with, the development of which was discussed previously in the development section on the report.

Development of the deliverable went reasonably smoothly, but could have been carried out better as some of the development work was carried out in a procedural coding fashion and had to later be converted in to an object oriented system, which did have a time penalty.

5.3. Areas for future development

Areas for future development of the system include turning the proof in to a working system, most likely using Java Servlets and JSPs along with PostgreSQL relational database system for data storage. Other obvious areas for future development include writing the modules for controlling other needed items such as:

- Email
- Databases
- FTP

Which are considered standard features with any hosting account these days.

To turn the system in to a commercial reality it would also need a GUI designing for it, for both the administrator and the client interfaces, ways of achieving this were discussed in the development section, with the conclusion being that most of it can be achieved by creating an XSLT style sheet to format the existing XML data in to an HTML form for editing, reducing the development of the GUI significantly. KDA Web Services Ltd. Have shown interest in taking these ideas and turning them in to a commercial product as have a few other hosting companies that were spoken too during the course of the project, which goes some way proving that there is possibility for commercial exploitation.

One area that was shown particular interest in was that of turning the use of the system to clustering, to provide redundant services, so that a web site may be set up on a cluster of web servers, rather than the traditional one server – so that if one web server fails, one of the others will take over. It would be prudent in the future to look further in to this issue as "normal" web hosting becomes abundant and commonplace web hosts need something to differentiate themselves from the rest of the market.

5.4. Conclusion

Conclusions that I have drawn from this project are that modular and flexible software can be achieved, but only if it is designed to be modular and flexible from the start, it is not something that can be bolted on afterwards as an afterthought to the design. It has to be carefully thought out, considering all aspects of the system, from storing configuration information for applications, handling communication between servers and the actual coding of the module interface.

From research carried out, I can also draw the conclusion that there is no current ideal hosting automation solution available to web hosts and there most likely never will be unless modular, pluggable software is developed for the market – The reason being that all web hosts have a different definition of ideal, for some it is Windows based hosting, for others it is being able to offer a combination of Linux and Windows based hosting with clustering support. No one product in its entirety is going to be ideal for everyone, as when you try to be the jack of all trades, you become master of none. So the solution to this is to produce a base product that has enough features to meat the common ground for all hosts, then provide a framework in to which hosts may add their own modules, to do their own specific tasks to tailor the system to their ideals. What it also means for hosts, is that they have a potential new revenue stream from marketing their own plug-in modules for the system, and if more modules/features for a product are on the market, then there is more incentive for people to buy the base product.

My final conclusion is that any company that has the vision, the time and the skills to take these ideas and develop a system with them, has the potential to be a market leader and reap the rewards that such a status brings.

References

cPanel Inc., (2003), CPanel.NET, [online], last accessed on 10 February 2003 at URL: http://www.cpanel.net

Dearden, Andrew, (2003, February 6th), Case Study 3: Patterns & the SWING libraries, CMS, SHU

Netcraft Ltd., (1996), Netcraft Web Server Survey - All Domains, [online], last accessed on 26 March 2003 at URL: http://www.netcraft.com/Survey/Reports/9601/ALL/

Netcraft Ltd., (2003), March 2003 Web Server Survey, [online], last accessed on 26 March 2003 at URL: http://news.netcraft.com/archives/2003/03/25/march_2003_web_server_survey.html

Positive Software Corporation, (2003), Positive Software Corporation, [online], last accessed on 10 February 2003 at URL: http://www.psoft.net

Rational Software, (2003), Unified Modeling Language Resource Center, [online], last accessed on 18 March 2003 at URL: http://www.rational.com/uml/index.jsp

Smartdraw, (2003), Learn about the six SSADM4 stages, [online], last accessed on 4 April 2003 at URL: http://www.smartdraw.com/resources/centers/software/ssadm.htm

W3C, (1999), XSL Transformations (XSLT), [online], last accessed on 2 April 2003 at URL: http://www.w3.org/TR/xslt

ZoneEdit.Com, (2002), Simplified example of how DNS works, [online], last accessed on 12 February 2003 at URL: http://www.zoneedit.com/doc/dns-basics.html

Bibliography

Apache Software Foundation, (2003), The Apache HTTP Server Project, [online], last accessed on 6 March 2003 at URL: http://httpd.apache.org

Bernstein, D. J., (2003), djbdns: Domain Name System tools, [online], last accessed on 8 March 2003 at URL: http://cr.yp.to/djbdns.html

Floyd, Michael, (2000), *Building Web Sites With XML*, Prentice Hall

Hillside.net, (2001), Patterns Home Page, [online], last accessed on 20 March 2003 at URL: http://www.hillside.net/patterns/

Internet Software Consortium, (2003), Internet Software Consortium – BIND, [online], last accessed on 8 March 2003 at URL: http://www.isc.org/products/BIND

Jaiman, Ashish, (2001), Abstract Factory Pattern, [online], last accessed on 4 April 2003 at URL: http://www.dotnetextreme.com/articles/abstractfactory.asp

Medvidovic , Nenad, (1999), Assessing the suitability of UML for modelling Software Architectures, [online], last accessed on 4 April 2003 at URL: http://www.ics.uci.edu/~irus/bart/flyers/99/presentations/july99-neno/

Pressman, Roger (2000), *Software Engineering, A Practitioner's Approach, 5th Edition, McGraw Hill*

Smartdraw, (2003), Types of UML Diagrams – Unified Modeling Language (UML), [online], last accessed on 4 April 2003 at URL: http://www.smartdraw.com/resources/centers/uml/uml.htm

W3C, (2001), XML Base, [online], last accessed on 21 February 2003 at URL: http://www.w3.org/TR/xmlbase/

Appendix A – Project Specification

Project Definition

Team: Karl Austin supervised by Mike Parr

Date: 10/11/2002

Degree Route: BSc (Hons) Software Engineering

Title: Developing modular software with reference to web hosting automation

Elaboration

Description

To produce a modular proof of concept deliverable implementing the web server and DNS sections of web host automation software based on the client-server architecture. The project aims to show that moderately platform independent, highly modular and scalable software can be developed for deployment by users with little knowledge of the application domain.

What it involves

Evaluate current solutions

Evaluate hardware solutions

Evaluate language for implementation

Evaluate communication protocols

Objectives and Deliverables

Refined Description

The system will demonstrate how robust, modular, scalable client-server applications can be developed with current technologies and development techniques. It will demonstrate the management of the Apache web server software and the BIND name server software – although as an alternative it will also demonstrate how

DJBDNS name server software could be used in place of BIND by writing a plug-in module for it.

What will be produced?

A proof of concept deliverable for the ideas discussed and will implement the code needed to add domains to a web server which will involve such things as creating user accounts, creating/ editing DNS server entries and creating/ editing entries in the Apache configuration files.

Tasks

Task	**Duration**
Research software requirements	8 Hours
Investigate existing solutions	24 Hours
Asses existing software against requirements	12 Hours
Create development plan	4 Hours
Investigate appropriate technology for implementation	60 Hours
System Design	24 Hours
Development	98 Hours
Critical Evaluation	24 Hours
Final Report	48 Hours
TOTAL	**300 Hours**

Marks Breakdown

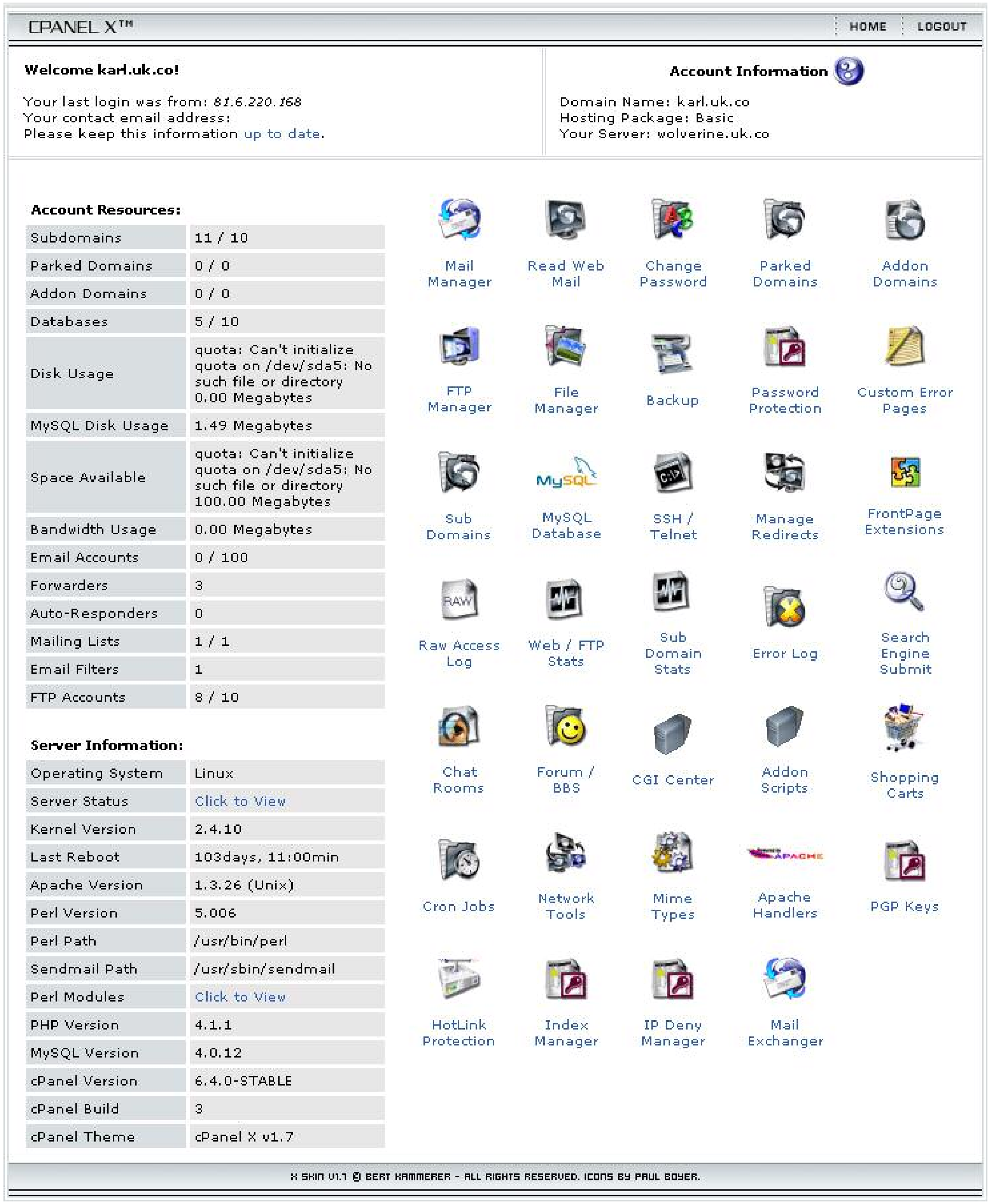
Process:

1. Investigation	20
2. Design	10
3. Development	10
	40% Total

Deliverable:

1.	Quality of description	15
2.	Quality of deliverable	15
		30% Total

Appendix B – User Interface Screen-shots



Welcome karl.uk.co!

Your last login was from: *81.6.220.168*
Your contact email address:
Please keep this information up to date.

Account Information

Domain Name: karl.uk.co
Hosting Package: Basic
Your Server: wolverine.uk.co

Account Resources:

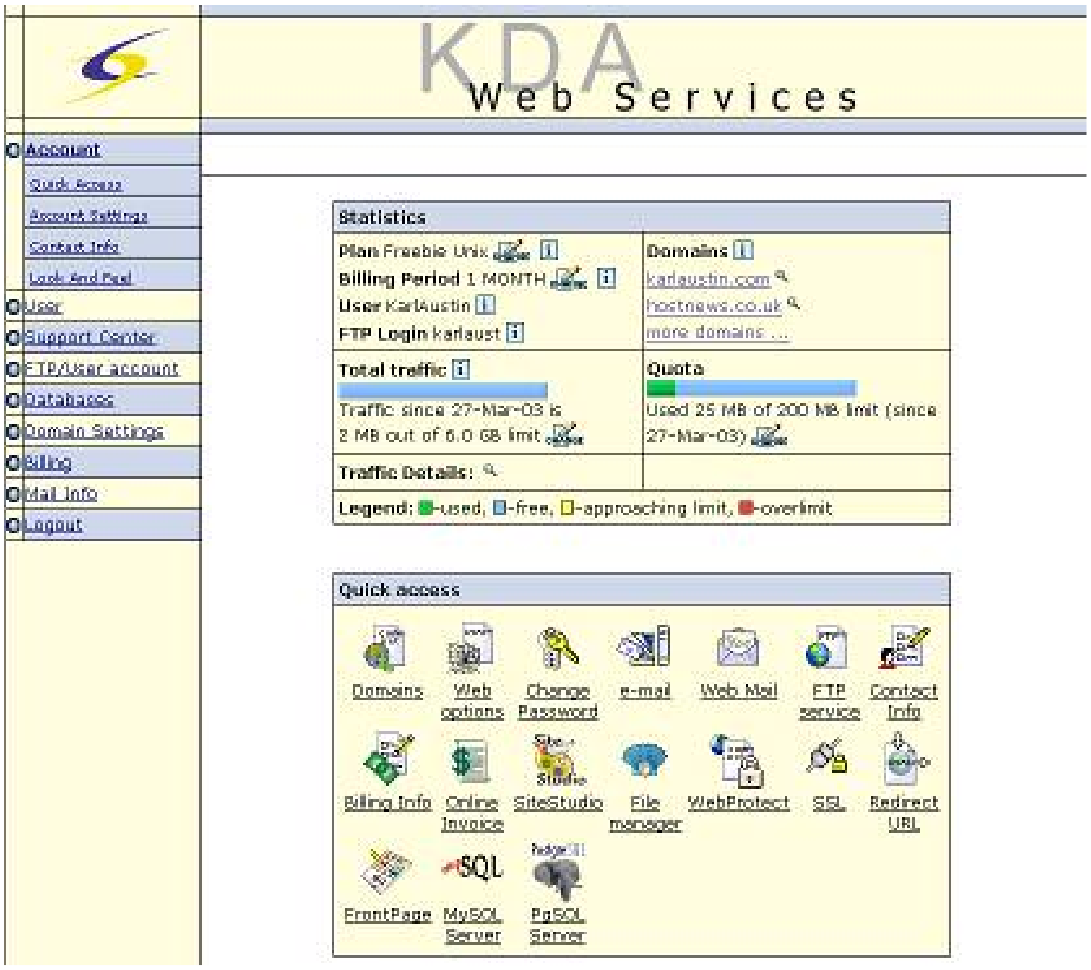
Subdomains	11 / 10
Parked Domains	0 / 0
Addon Domains	0 / 0
Databases	5 / 10
Disk Usage	quota: Can't initialize quota on /dev/sda5: No such file or directory 0.00 Megabytes
MySQL Disk Usage	1.49 Megabytes
Space Available	quota: Can't initialize quota on /dev/sda5: No such file or directory 100.00 Megabytes
Bandwidth Usage	0.00 Megabytes
Email Accounts	0 / 100
Forwarders	3
Auto-Responders	0
Mailing Lists	1 / 1
Email Filters	1
FTP Accounts	8 / 10

Server Information:

Operating System	Linux
Server Status	Click to View
Kernel Version	2.4.10
Last Reboot	103days, 11:00min
Apache Version	1.3.26 (Unix)
Perl Version	5.006
Perl Path	/usr/bin/perl
Sendmail Path	/usr/sbin/sendmail
Perl Modules	Click to View
PHP Version	4.1.1
MySQL Version	4.0.12
cPanel Version	6.4.0-STABLE
cPanel Build	3
cPanel Theme	cPanel X v1.7

Figure 4: CPanel End User GUI

Karl Austin



Statistics

Plan Freebie Unix
Billing Period 1 MONTH
User KarlAustin
FTP Login karlaust

Domains
karlaustin.com
hostnews.co.uk
more domains ...

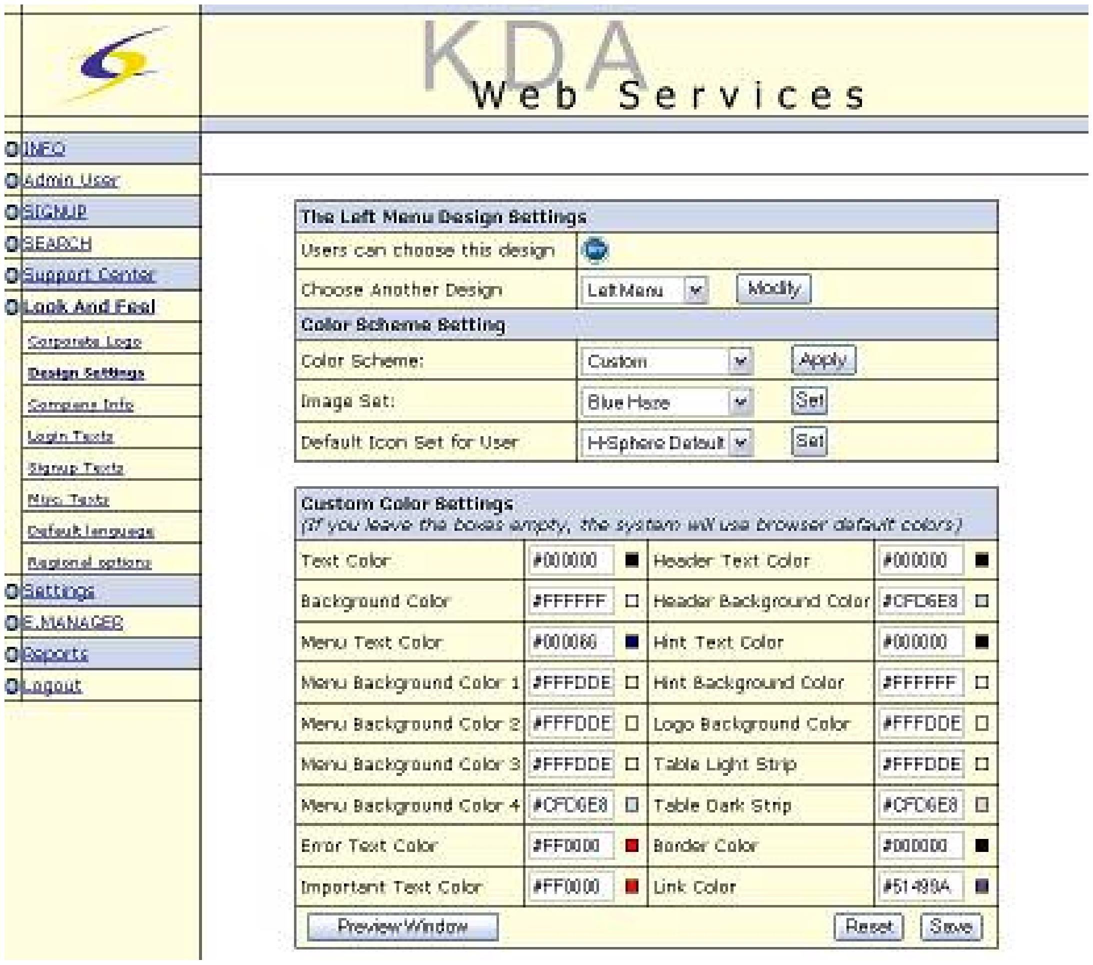
Total traffic
Traffic since 27-Mar-03 is 2 MB out of 6.0 GB limit

Quota
Used 25 MB of 200 MB limit (since 27-Mar-03)

Traffic Details:

Legend: 🟩-used, 🟦-free, 🟨-approaching limit, 🟥-overlimit

Figure 5: HSphere End User GUI


KDA Web Services

The Left Menu Design Settings	
Users can choose this design	●
Choose Another Design	LeftMenu [Modify]
Color Scheme Setting	
Color Scheme:	Custom [Apply]
Image Set:	Blue Haze [Set]
Default Icon Set for User	H-Sphere Default [Set]

Custom Color Settings
(If you leave the boxes empty, the system will use browser default colors)

Text Color	#000000	■	Header Text Color	#000000	■
Background Color	#FFFFFF	□	Header Background Color	#CFD6E8	□
Menu Text Color	#000060	■	Hint Text Color	#000000	■
Menu Background Color 1	#FFFDDE	□	Hint Background Color	#FFFFFF	□
Menu Background Color 2	#FFFDDE	□	Logo Background Color	#FFFDDE	□
Menu Background Color 3	#FFFDDE	□	Table Light Strip	#FFFDDE	□
Menu Background Color 4	#CFD6E8	□	Table Dark Strip	#CFD6E8	□
Error Text Color	#FF0000	🟥	Border Color	#000000	■
Important Text Color	#FF0000	🟥	Link Color	#51499A	■

Figure 6: HSphere Skin Colour Chooser

Appendix C – General System Data

Domain Configuration Data

```
<website id="10001" ip="62.149.37.18" port="80" server="62.149.37.18"
suexec="off">
                            <user>karl</user>
                           <group>karl</group>
          <root>/meta/data/web/karl/test.karlaustin.com</root>
                           <domain prefix="*">
                                    <primary>test.karlaustin.com</primary>
                                           <alias>messingham.org</alias>
                                </domain>
                                  <log>
                                    <format name="common">%h %l %u %t \"%r\" %>s %b
                                    <format name="referer">%{Referer}i -> %U</forma
                                    <format name="agent">%{User-agent}i</format>
                                    <error status="on">|/usr/bin/cronolog /meta/dat
                                    <custom format="combined">|/usr/bin/cronolog /m
                                 </log>
                               <errorpage>
                                    <page code="404">http://test.karlaustin.com/404
                              </errorpage>
                           <gzip status="on"/>
                            <cgi status="on">
                                             <extension>.cgi</extension>
                                              <extension>.pl</extension>
                                 </cgi>
                 <php status="on" restrictions="level1">
                                             <extension>.php</extension>
                                            <extension>.php3</extension>
                                            <extension>.php4</extension>
                                           <extension>.phtml</extension>
                                 </php>
                            <ssi status="on">
                                           <extension>.shtml</extension>
                                            <extension>.shtm</extension>
                                 </ssi>
                          <index type="fancy">
                                                       <file>.??*</file>
                                                         <file>*~</file>
                                                         <file>*#</file>
                                                    <file>HEADER*</file>
                                                    <file>README*</file>
                                                        <file>RCS</file>
                                                        <file>CVS</file>
                                                        <file>*,v</file>
                                                        <file>*,t</file>
                                </index>
</website>
```

DNS Configuration Data

```
<domain name="karlaustin.com" ttl="14400" ip="62.149.37.15"
reverse="yes" wildcard="yes" tld="com" sld="karlaustin">
      <serial>2002111317</serial>
      <refresh>10800</refresh>
      <retry>3600</retry>
      <expire>604800</expire>
      <ttl>86400</ttl>
      <soa>rogue1.kdawebservices.com</soa>
      <hostmaster>hostmaster.karlaustin.net</hostmaster>
      <mail priority="10" ttl="14400">mail.karlaustin.com</mail>
      <mail priority="20" ttl="14400">mail2.karlaustin.com</mail>
      <nameserver ttl="14400" type="primary"
ip="62.149.37.18">rogue1.kdawebservices.com</nameserver>
      <nameserver ttl="14400" type="secondary"
ip="62.149.37.18">rogue2.kdawebservices.com</nameserver>
      <cname ttl="14400" name="kda">www.kdawebservices.com</cname>
      <cname ttl="14400" name="home">karl.gotdns.com</cname>
      <subdomain name="uni" ttl="14400" ip="62.149.37.15" reverse="no"
wildcard="yes">
            <mail priority="10" ttl="14400">mail.karlaustin.com</mail>
            <mail priority="20" ttl="14400">mail2.karlaustin.com</mail>
            <cname ttl="14400" name="shu">www.shu.ac.uk</cname>
      </subdomain>
</domain>
```

Appendix D – XSLT + Output

Domain Configuration

Apache - XSL

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

      <xsl:output method="text"/>
      <xsl:include href="config.variables.xsl"/>

      <xsl:template match="website">
      <xsl:if test="@port!='80'">
&lt;VirtualHost <xsl:value-of select="@ip"/>:<xsl:value-of
select="@port"/>&gt;
      </xsl:if>
      <xsl:if test="@port='80'">
&lt;VirtualHost <xsl:value-of select="@ip"/>&gt;
      </xsl:if>
      <xsl:if test="suexec='on'">
      User <xsl:value-of select="user"/>
      Group <xsl:value-of select="group"/>
      </xsl:if>
      DocumentRoot <xsl:value-of select="root"/>
      <xsl:apply-templates select="domain"/>
      <xsl:apply-templates select="log"/>
      <xsl:apply-templates select="errorpage"/>
      <xsl:if test="cgi/@status='on'">
            <xsl:apply-templates select="cgi"/>
      </xsl:if>
      <xsl:if test="gzip/@status='on'">
            <xsl:apply-templates select="gzip"/>
      </xsl:if>
      <xsl:apply-templates select="index"/>
      <xsl:if test="php/@status='on'">
            <xsl:apply-templates select="php"/>
      </xsl:if>
      <xsl:if test="ssi/@status='on'">
            <xsl:apply-templates select="ssi"/>
      </xsl:if>
      <xsl:if test="ssl/@status='on'">
            <xsl:apply-templates select="ssl"/>
      </xsl:if>
&lt;/VirtualHost&gt;
      </xsl:template>


      <xsl:template match="domain">
      ServerName <xsl:value-of select="primary"/>
      ServerAlias<xsl:for-each select="alias|primary"><xsl:text>
</xsl:text><xsl:value-of select="../@prefix"/>.<xsl:value-of
select="."/></xsl:for-each>
      </xsl:template>
      <xsl:template match="log">
      LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-
Agent}i\"" combined
            <xsl:for-each select="format">
      LogFormat "<xsl:value-of select="."/>"<xsl:text>
</xsl:text><xsl:value-of select="@name"/>
            </xsl:for-each>
            <xsl:if test="error/@status='on'">
      ErrorLog "<xsl:value-of select="error"/>"
            </xsl:if>
            <xsl:for-each select="custom">
      CustomLog "<xsl:value-of select="."/>"<xsl:text>
</xsl:text><xsl:value-of select="@format"/>
            </xsl:for-each>
      </xsl:template>


      <xsl:template match="errorpage">
            <xsl:if test="count(page) > 0">
                  <xsl:for-each select="page">
      ErrorDocument <xsl:value-of select="@code"/><xsl:text>
</xsl:text><xsl:value-of select="."/>
                  </xsl:for-each>
            </xsl:if>
      </xsl:template>


      <xsl:template match="cgi">
      ScriptAlias /cgi-bin/ <xsl:value-of select="../root"/>
            <xsl:for-each select="extension">
      AddHandler cgi-script <xsl:value-of select="."/>
            </xsl:for-each>
      </xsl:template>


      <xsl:template match="gzip">

<xsl:text>
</xsl:text>

      </xsl:template>


      <xsl:template match="index">
      <xsl:if test="@type='fancy'"> IndexOptions FancyIndexing</xsl:if>
      IndexIgnore<xsl:for-each select="file"><xsl:text>
</xsl:text><xsl:value-of select="."/></xsl:for-each>
      </xsl:template>


      <xsl:template match="php">
            <xsl:choose>
                  <xsl:when test="@restrictions='level1'">
      php_admin_value open_basedir "<xsl:value-of
select="../root"/>:<xsl:value-of select="$PHP_PATH_PEAR"/>
```

```
                  </xsl:when>
                  <xsl:when test="@restrictions='level2'">
      php_admin_value open_basedir "<xsl:value-of
select="../root"/>:<xsl:value-of select="$PHP_PATH_PEAR"/>"
      php_admin_value safe_mode true
      php_admin_value safe_mode_include_dir <xsl:value-of
select="$PHP_PATH_PEAR"/>
                  </xsl:when>
            </xsl:choose>
            <xsl:for-each select="extension">
      AddType application/x-httpd-php <xsl:value-of select="."/>
            </xsl:for-each>
      AddType application/x-httpd-php-source .phps
      </xsl:template>


      <xsl:template match="ssi">
            <xsl:for-each select="extension">
      AddType text/html <xsl:value-of select="."/>
      AddHandler server-parsed <xsl:value-of select="."/>
            </xsl:for-each>
      </xsl:template>


      <xsl:template match="ssl">

<xsl:text>
</xsl:text>

      </xsl:template>


</xsl:stylesheet>
```

Apache - Output

```
<VirtualHost 62.149.37.18>

      DocumentRoot /meta/data/web/karl/test.karlaustin.com
      ServerName test.karlaustin.com
      ServerAlias *.test.karlaustin.com *.messingham.org
      LogFormat \"%h %l %u %t \\\"%r\\\" %>s %b \\\"%{Referer}i\\\"
\\\"%{User-Agent}i\\\"\" combined

      LogFormat \"%h %l %u %t \\\"%r\\\" %>s %b\" common
      LogFormat \"%{Referer}i -> %U\" referer
      LogFormat \"%{User-agent}i\" agent
      ErrorLog \"|/usr/bin/cronolog
/meta/data/web/karl/logs/test.karlaustin.com/error.log.%Y%m%d\"

      CustomLog \"|/usr/bin/cronolog
/meta/data/web/karl/logs/test.karlaustin.com/access.log.%Y%m%d\"
combined
      ErrorDocument 404 http://test.karlaustin.com/404.php
      ScriptAlias /cgi-bin/ /meta/data/web/karl/test.karlaustin.com
      AddHandler cgi-script .cgi
      AddHandler cgi-script .pl
      IndexOptions FancyIndexing
      IndexIgnore .??* *~ *# HEADER* README* RCS CVS *,v *,t
      php_admin_value open_basedir
\"/meta/data/web/karl/test.karlaustin.com:/usr/local/lib/php/
      AddType application/x-httpd-php .php
      AddType application/x-httpd-php .php3
      AddType application/x-httpd-php .php4
      AddType application/x-httpd-php .phtml
      AddType application/x-httpd-php-source .phps

      AddType text/html .shtml
      AddHandler server-parsed .shtml
      AddType text/html .shtm
      AddHandler server-parsed .shtm
</VirtualHost>
```

DNS Configuration

Bind – XSL

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

      <xsl:output method="text"/>

      <xsl:template match="domain">
$ORIGIN <xsl:value-of select="@tld"/>.
<xsl:value-of select="@sld"/><xsl:text>   </xsl:text><xsl:value-of
select="@ttl"/>   IN    A     <xsl:value-of select="@ip"/><xsl:text>
      </xsl:text>
      <xsl:value-of select="@ttl"/> IN SOA <xsl:value-of
select="soa"/>.<xsl:text> </xsl:text><xsl:value-of
select="hostmaster"/>. (
      <xsl:value-of select="serial"/><xsl:text> </xsl:text><xsl:value-
of select="refresh"/><xsl:text> </xsl:text><xsl:value-of
select="retry"/><xsl:text> </xsl:text><xsl:value-of
select="expire"/><xsl:text> </xsl:text><xsl:value-of select="ttl"/> )

<xsl:for-each select="nameserver">
      <xsl:text>  </xsl:text><xsl:value-of select="@ttl"/>  IN    NS
      <xsl:value-of select="."/>.<xsl:text>
</xsl:text>
      </xsl:for-each>
      <xsl:for-each select="mail">
<xsl:text>  </xsl:text><xsl:value-of select="@ttl"/>  IN    MX
      <xsl:value-of select="@priority"/><xsl:text>
</xsl:text><xsl:value-of select="."/>.<xsl:text>
</xsl:text>
      </xsl:for-each>
$ORIGIN <xsl:value-of select="@name"/>.
<xsl:if test="@wildcard='yes'">*    <xsl:value-of select="@ttl"/> IN
      A     <xsl:value-of select="@ip"/><xsl:text>
</xsl:text></xsl:if>

      <xsl:for-each select="mail">
      <xsl:if test="../@wildcard='yes'">* <xsl:value-of select="@ttl"/>
      IN    MX    <xsl:value-of select="@priority"/><xsl:text>
</xsl:text><xsl:value-of select="."/>.
</xsl:if>
      </xsl:for-each>

      <xsl:for-each select="cname">
<xsl:value-of select="@name"/><xsl:text>  </xsl:text><xsl:value-of
select="@ttl"/>   IN    CNAME <xsl:value-of select="."/>.<xsl:text>
</xsl:text>
      </xsl:for-each>

      <xsl:for-each select="subdomain">
<xsl:value-of select="@name"/><xsl:text>  </xsl:text><xsl:value-of
select="@ttl"/>   IN    A     <xsl:value-of select="@ip"/><xsl:text>
</xsl:text>
```

```
      <xsl:for-each select="mail">
      <xsl:text>  </xsl:text><xsl:value-of select="@ttl"/>  IN    MX
      <xsl:value-of select="@priority"/><xsl:text>
</xsl:text><xsl:value-of select="."/>.<xsl:text>
</xsl:text>
      </xsl:for-each>
      </xsl:for-each>

      <xsl:for-each select="subdomain">
$ORIGIN <xsl:value-of select="@name"/>.<xsl:value-of
select="../@name"/>.<xsl:text>
</xsl:text>
<xsl:if test="../@wildcard='yes'">* <xsl:value-of select="@ttl"/> IN
      A     <xsl:value-of select="@ip"/><xsl:text>
</xsl:text></xsl:if>
      <xsl:for-each select="mail">
      <xsl:if test="../@wildcard='yes'">* <xsl:value-of select="@ttl"/>
      IN    MX    <xsl:value-of select="@priority"/><xsl:text>
</xsl:text><xsl:value-of select="."/>.
</xsl:if>
      </xsl:for-each>
      <xsl:for-each select="cname">
<xsl:value-of select="@name"/><xsl:text>  </xsl:text><xsl:value-of
select="@ttl"/>   IN    CNAME <xsl:value-of select="."/>.<xsl:text>
</xsl:text>
      </xsl:for-each>
      </xsl:for-each>

      </xsl:template>

</xsl:stylesheet>
```

Bind – Output

```
$ORIGIN com.
karlaustin  14400 IN    A     62.149.37.15
      14400 IN SOA rogue1.kdawebservices.com.
hostmaster.karlaustin.net. (
      2002111317 10800 3600 604800 86400 )

      14400 IN    NS    rogue1.kdawebservices.com.
      14400 IN    NS    rogue2.kdawebservices.com.
      14400 IN    MX    10 mail.karlaustin.com.
      14400 IN    MX    20 mail2.karlaustin.com.

$ORIGIN karlaustin.com.
*     14400 IN    A     62.149.37.15
*     14400 IN    MX    10 mail.karlaustin.com.
*     14400 IN    MX    20 mail2.karlaustin.com.
kda   14400 IN    CNAME www.kdawebservices.com.
home  14400 IN    CNAME karl.gotdns.com.
uni   14400 IN    A     62.149.37.15
      14400 IN    MX    10 mail.karlaustin.com.
      14400 IN    MX    20 mail2.karlaustin.com.

$ORIGIN uni.karlaustin.com.
```

```
*     14400 IN    A     62.149.37.15
*     14400 IN    MX    10 mail.karlaustin.com.
*     14400 IN    MX    20 mail2.karlaustin.com.
shu   14400 IN    CNAME www.shu.ac.uk.
```

DJBDNS – XSL

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

      <xsl:output method="text"/>

      <xsl:template match="domain">
#START#<xsl:value-of select="@name"/>#
Z<xsl:value-of select="@name"/>:<xsl:value-of
select="soa"/>:<xsl:value-of select="hostmaster"/>:<xsl:value-of
select="serial"/>:<xsl:value-of select="refresh"/>:<xsl:value-of
select="retry"/>:<xsl:value-of select="expire"/>::<xsl:value-of
select="ttl"/>
      <xsl:choose>
            <xsl:when test="@reverse='yes'">
=<xsl:value-of select="@name"/>:<xsl:value-of
select="@ip"/>:<xsl:value-of select="@ttl"/>
            </xsl:when>
            <xsl:otherwise>
+<xsl:value-of select="@name"/>:<xsl:value-of
select="@ip"/>:<xsl:value-of select="@ttl"/>
            </xsl:otherwise>
      </xsl:choose>
      <xsl:if test="@wildcard='yes'">
+*.<xsl:value-of select="@name"/>:<xsl:value-of
select="@ip"/>:<xsl:value-of select="@ttl"/>
      </xsl:if>

      <xsl:for-each select="nameserver">
.<xsl:value-of select="../@name"/>::<xsl:value-of
select="."/>:<xsl:value-of select="@ttl"/>
      </xsl:for-each>

      <xsl:for-each select="mail">
@<xsl:value-of select="../@name"/>::<xsl:value-of
select="."/>:<xsl:value-of select="@priority"/>:<xsl:value-of
select="@ttl"/>
            <xsl:if test="../@wildcard='yes'">
@*.<xsl:value-of select="../@name"/>::<xsl:value-of
select="."/>:<xsl:value-of select="@priority"/>:<xsl:value-of
select="@ttl"/>
            </xsl:if>
      </xsl:for-each>

      <xsl:for-each select="cname">
C<xsl:value-of select="@name"/>.<xsl:value-of
select="../@name"/>:<xsl:value-of select="."/>:<xsl:value-of
select="@ttl"/>
      </xsl:for-each>
      <xsl:for-each select="subdomain">
      <xsl:choose>
            <xsl:when test="@reverse='yes'">
=<xsl:value-of select="@name"/>.<xsl:value-of
select="../@name"/>:<xsl:value-of select="@ip"/>:<xsl:value-of
select="@ttl"/>
            </xsl:when>
            <xsl:otherwise>
+<xsl:value-of select="@name"/>.<xsl:value-of
select="../@name"/>:<xsl:value-of select="@ip"/>:<xsl:value-of
select="@ttl"/>
            </xsl:otherwise>
      </xsl:choose>
      <xsl:if test="@wildcard='yes'">
+*.<xsl:value-of select="@name"/>.<xsl:value-of
select="../@name"/>:<xsl:value-of select="@ip"/>:<xsl:value-of
select="@ttl"/>
      </xsl:if>

      <xsl:for-each select="mail">
@<xsl:value-of select="../@name"/>.<xsl:value-of
select="../../@name"/>::<xsl:value-of select="."/>:<xsl:value-of
select="@priority"/>:<xsl:value-of select="@ttl"/>
      <xsl:if test="../@wildcard='yes'">
@*.<xsl:value-of select="../@name"/>.<xsl:value-of
select="../../@name"/>::<xsl:value-of select="."/>:<xsl:value-of
select="@priority"/>:<xsl:value-of select="@ttl"/>
      </xsl:if>
      </xsl:for-each>

      <xsl:for-each select="cname">
C<xsl:value-of select="@name"/>.<xsl:value-of
select="../@name"/>.<xsl:value-of select="../../@name"/>:<xsl:value-of
select="."/>:<xsl:value-of select="@ttl"/>
      </xsl:for-each>
      </xsl:for-each>
#END#<xsl:value-of select="@name"/>#
      </xsl:template>

</xsl:stylesheet>
```

DJBDNS – Output

```
#START#karlaustin.com#
Zkarlaustin.com:rogue1.kdawebservices.com:hostmaster.karlaustin.net:200
2111317:10800:3600:604800::86400
=karlaustin.com:62.149.37.15:14400
+*.karlaustin.com:62.149.37.15:14400
.karlaustin.com::rogue1.kdawebservices.com:14400
.karlaustin.com::rogue2.kdawebservices.com:14400
@karlaustin.com::mail.karlaustin.com:10:14400
@*.karlaustin.com::mail.karlaustin.com:10:14400
@karlaustin.com::mail2.karlaustin.com:20:14400
@*.karlaustin.com::mail2.karlaustin.com:20:14400
Ckda.karlaustin.com:www.kdawebservices.com:14400
Chome.karlaustin.com:karl.gotdns.com:14400
```

```
+uni.karlaustin.com:62.149.37.15:14400
+*.uni.karlaustin.com:62.149.37.15:14400
@uni.karlaustin.com::mail.karlaustin.com:10:14400
@*.uni.karlaustin.com::mail.karlaustin.com:10:14400
@uni.karlaustin.com::mail2.karlaustin.com:20:14400
@*.uni.karlaustin.com::mail2.karlaustin.com:20:14400
Cshu.uni.karlaustin.com:www.shu.ac.uk:14400
#END#karlaustin.com#
```

Appendix E – Application Code

ApacheWebServer.php

```
<?php

                      include_once 'WebServer.php';

                 class ApacheWebServer extends WebServer
                                   {

                                                    function ApacheWebServer()
                                                                            {

                                                                                   echo
                                                                                   paren

                                                                            }

                                                   function runConfiguration()
                                                                            {

                                                                                   echo
                                                                                   paren
                                                                                   $this

                                                                            }

                                   }

?>
```

BindDNSServer.php

```
<?php

                      include_once 'DNSServer.php';

                  class BindDNSServer extends DNSServer
                                   {

                                                      function BindDNSServer()
                                                                            {

                                                                                   echo
                                                                                   paren

                                                                            }

                                                   function runConfiguration()
                                                                            {

                                                                                   echo
                                                                                   paren
                                                                                   $this

                                                                            }

                                   }
```

```
?>
```

DJBDNSServer.php

```
<?php
                              include_once 'DNSServer.php';

                         class DJBDNSServer extends DNSServer
                                                 {
                                                                 function DJBDNSServer()
                                                                                               {

                                                                                                        echo
                                                                                                        paren

                                                                                               }

                                                             function runConfiguration()
                                                                                               {

                                                                                                        echo
                                                                                                        paren
                                                                                                        $this

                                                                                               }

                                                 }
?>
```

DNSServer.php

```
<?php
                                include_once 'Server.php';

                             class DNSServer extends Server
                                                 {
                                                             function DNSServer( $name )
                                                                                               {

                                                                                                        echo

                                                                                               }

                                                          function _constructor( $name )
                                                                                               {

                                                                                                        echo
                                                                                                        paren

                                                                                               }

                                                    function _parseConfigData( $vars = '' )
                                                                                               {
```

```
                                                                                echo
                                                                                paren

                                                                          }

                                                 }

?>
```

DNSServerFactory.php

```
<?php

                          class DNSServerFactory
                                                 {

                                                          function DNSServerFactory()
                                                                          {

                                                                                echo

                                                                          }

                                                       function createServer( $type )
                                                                          {

                                                                                echo
                                                                                inclu
                                                                                $clas
                                                                                retur

                                                                          }

                                                 }

?>
```

Server.php

```
<?php

                      include_once 'Utilities.php';

                               class Server
                                                 {

                                                                var $name = '';
                                                               var $files = '';
                                                            var $commands = '';

                                                            var $_address = '';
                                                      var $_configuration = '';
                                                         var $_execOutput = '';
                                                                  var $_os = '';
                                                            var $_replace = '';

                                                               var $utils = '';

                                                              function Server()
```

```
                                                  {
                                                          echo
                                                  }
                              function _constructor( $name, $os )
                                                  {
                                                          echo
                                                          $this
                                                          $this
                                                          $this
                                                  }
                                function execCommand( $command )
                                                  {
                                                          echo
                                                          $comm
                                                          retur
                                                  }
                                   function runConfiguration()
                                                  {
                                                          $this
                                                  }
                               function setConfiguration( $xml )
                                                  {
                                                          echo
                                                          $this
                                                  }
                                function _setAddress( $address )
                                                  {
                                                          echo
                                                          $this
                                                  }
                            function _setReplace( $name, $value )
                                                  {
                                                          echo
                                                          $this
                                                  }
                              function __execCommand( $command )
                                                  {
                                                          echo
                                                          $cmd
```

```
                                                                                        exec(
                                                                                        $this
                                                                                        echo
                                                                                        retur

                                                                                  }

                                                    function _parseConfigData( $type )
                                                                                  {

                                                                                        echo

                                                                                        $xml
                                                                                        $xsl

                                                                                        $xml
                                                                                        $outp

                                                                                        eval(
                                                                                        $this
                                                                                        $this

                                                                                  }

                                                  function _processFileConfiguration( $command )
                                                                                  {

                                                                                        echo

                                                                                        while
















then '.$writerLine.'
fi';
















                                                                                        }

                                                                                  }

                                                  }
```

```
?>
```

ServerFactory.php

```
<?php

                           class ServerFactory
                                    {

                                                      function ServerFactory()
                                                                            {

                                                                                   echo

                                                                            }

                                               function createFactory( $type )
                                                                            {

                                                                                   echo
                                                                                   inclu
                                                                                   $clas
                                                                                   retur

                                                                            }

                                    }

?>
```

Utilities.php

```
<?php

                             class Utilities
                                    {

                                                          function Utilities()
                                                                            {
                                                                                   echo
                                                                            }

                                            function domxml_xmlarray ($branch)
                                                                            {
                                                                                   echo

                                                                                   $obje
                                                                                   $objp
                                                                                   $bran

                                                                                   while
```

```
                                                                    }
                                                                    retur
                                                             }

                                       function getFileContents( $file )
                                                             {

                                                                    echo

                                                                    retur

                                                             }

                                     function replaceVars( $text, $vars ) {

                                                                    echo

                                                                    while



                                                                    }

                                                                    retur

                                                             }

                                      function transformXML( $xml, $xsl )
                                                             {

                                                                    echo

                                                                    $args
                                                                    $x =
                                                                    xslt_
                                                                    $outp
                                                                    if (!

                                                                    }
                                                                    xslt_
                                                                    retur

                                                             }

                                     }
?>
```

WebServer.php

```
<?php
                       include_once 'Server.php';

                     class WebServer extends Server
                                    {
                                                    function WebServer()
                                                                       {

                                                                              echo

                                                                       }

                                                 function parseConfiguration()
                                                                       {

                                                                              echo

                                                                              $xsl
                                                                              $ret
                                                                              paren
                                                                              retur

                                                                       }

                                                   function runConfiguration()
                                                                       {

                                                                              echo

                                                                              $this
                                                                              $this
                                                                              paren

                                                                       }

                                             function setConfiguration( $xml )
                                                                       {

                                                                              echo

                                                                              $dom
                                                                              $deta

                                                                              $serv
                                                                              $id =

                                                                              paren
                                                                              paren
                                                                              paren

                                                                       }

                                           function _constructor( $name, $os )
                                                                       {

                                                                              echo
                                                                              paren

                                                                       }

                                                   function _parseConfigData()
                                                                       {
```

```
                                                                                             echo
                                                                                             paren
                                                                                      }
                                             }
?>
```

WebserverFactory.php

```
<?php
                         class WebServerFactory
                                   {
                                            function WebServerFactory()
                                                                      {
                                                                             echo
                                                                      }
                                         function createServer( $type )
                                                                      {
                                                                             echo
                                                                             inclu
                                                                             $clas
                                                                             retur
                                                                      }
                                   }
?>
```

Karl Austin