Performance measurement as a tool for Software Engineering

by

JAN MARKUS VAN AARDT

Submitted in fulfilment of part of the requirements for the degree of

MAGISTER SCIENTIAE

in the faculty of

NATURAL AND AGRICULTURAL SCIENCES

University of Pretoria

SUPERVISOR: PROF. B.W. WATSON

November 2002

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

ACKNOWLEDGEMENTS

I would like to thank the following people and instances:

My supervisor, Prof. Bruce Watson, for his excellent guidance

My employer, Deutsche Bank, for study leave and the provision of software

My brothers and sister for their continuous support and inspiration

My loving girlfriend for countless sacrifices and endless support

My parents for endless faith and love

My heavenly Father for grace and love beyond measure

Contents

List of tables

List of figures

List of appendices

Abstract

Some software development teams regard software performance measurement as a mere luxury. When it happens, it often tends to be infrequent, insufficient and subjective. Countless software projects were sent into an uncontrollable spiral of poor management and unsatisfactory results. By revisiting old ideas and policies, many companies have turned themselves around. To ensure that software engineering does the same, technologies and procedures have to be reevaluated. The fact that many companies have decided to cut costs on technology expenditure necessitates software development teams to look for alternative options for deploying high performance software systems. As many companies are moving into the electronic era and evolving to the next stage of evolution, electronic commerce, the more important it has become to apply these concepts on Internet development projects and procedures.

The Internet market has shown that two software providers are aiming for worldwide domination of Internet server deployment, being Microsoft and Apache. Currently, the Apache web server is the most commonly used server on the Internet today (60%), with Microsoft's Internet Information Server (25%) in a strong second place. The need for higher throughput and better services is getting more with each passing day. It increases the pressure on these two software vendors to provide the best architecture for their clients' needs. This study intends to provide the reader with an objective view of a basic performance comparison between these two products and tries to find a correlation between the performance tests and the products' popularity standings. The tests for this study were performed on identical hardware architectures with one difference, being the operating system. By comparing the costly proprietary Microsoft solution with its cheaper open source rival, Linux, certain opinions were tested. Would a product developed by a software company that invests millions of dollars in their products perform better than this free-for-all solution, or would the selfless inputs of hundreds of developers all over the world finally pay off through the creation of the world's best Internet server?

The results of these tests were evaluated through formal statistical methods, providing overall comparisons of several common uses of web servers. These results were implemented in a small field test to prove the findings in practice with some interesting outcomes in terms of supportive technologies, new rapid application development (RAD) tools and data access models.

Abstract

This research in itself will not change the mind of any Internet programmer. What it hopes to achieve is to demonstrate software engineers that current processes and methods of developing software are not always the right way of doing things. Furthermore, it highlights many important factors often ignored or overlooked while managing software projects. Change management, process re-engineering and risk management form crucial elements of software development projects. By not adhering to certain critical elements of software development, software projects stand the chance of not reaching their goals and could even fail completely. Performance measurement acts as a tool for software engineering, providing guidelines for technological decisions, project management and ultimately, project success.

Ekserp

Die belangrikheid van prestasiemeting in die ontwikkeling van sagteware word nie altyd na waarde geskat nie. Programmeringspanne verkies dit om hierdie stap in die sagteware ontwikkelings proses te ontwyk, of dit af te water net om korporatiewe vereistes na te kom. Vele sagteware projekte het al gesneuwel as gevolg van swak bestuur en die onbereikbaarheid van projek doelwitte. Baie maatskappye het hulself op 'n pad van hernuwing geplaas deur die hersiening van korporatiewe prosedures en beleidstellings. Dieselfde hernuwing is krities om te verseker dat programmatuur-ingenieurswese aan die voorpunt bly deur die herevaluering van tegnologieë en prosedures. Wat die situasie soveel meer dringend maak vir die inligtingstegnologie werksmag is die feit dat kostebesparings deur besighede hulle gaan noodsaak om goedkoper alternatiewe te oorweeg. Die opvlamming van elektroniese handel forseer dieselfde argumente af op Internet ontwikkelings prosesse.

Die Internet se huidige beeld wys dat twee Internet bedienerstelsels die mark oorheers. Microsoft en Apache is die twee voorste verskaffers van Internet bediening sagteware en hou tans vyf-en-twintig en sestig persent onderskeidelik van die mark. As gevolg van die Internet se eksponensiële groei gaan sagteware verskaffers onder meer druk begin verkeer om vinniger en meer doeltreffende stelsels te verskaf. Hierdie studie poog om die leser te lei deur 'n objektiewe prestasie evalueringsproses om te vind of daar 'n verband bestaan tussen die huidige mark aandeel van die onderskeie Internet bedieners en die sagteware se prestasie en funksionaliteit. Hierdie toetse was uitgevoer op 'n identiese hardeware raamwerk met een verskil: die bedryfstelsel. Deur die duur Microsoft Windows Server met die goedkoper alternatief, Linux, te vergelyk, kry die leser ook insig in die verskille tussen vry-kode ontwikkeling teenoor Microsoft se kommersiële ontwikkelingsbenadering. Die vraag kan dan ook gevra word of kommersiële sagteware, met die rugsteuning van miljoene dollars beter sal vaar as vry-kode sagteware wat hoofsaaklik deur filantropiese programmeerders ontwikkel word.

Die resultate wat verkry is uit hierdie toetse word verder deur statistiese metodes ontleed om statisties beduidende verskille te identifiseer. Verder word die inligting gebruik in die ontwikkeling van 'n korporatiewe Internet stelsel om dit in die praktyk toe te pas. Interessante bevindings was ook gemaak in terme van ondersteunstegnologieë en die gebruik van ontwikkelingsgereedskap (RAD tools) en data toegangstegnologieë op beide platforms.

Die doel van hierdie navorsing is nie daarop gemik om sagteware ontwikkelaars te forseer om van tegnologieë te verander nie. Wat graag bereik wil word hierdeur is om die leser bewus te maak van die belangrikheid van proseshersiening ten einde beter produkte te kan vervaardig. Dit lig verder ook belangrike faktore uit wat maklik geïgnoreer word in die bestuur van sagteware ontwikkelingsprosesse. Veranderingsbestuur (change management), proses heringenieuring (process re-engineering) en risikobestuur vorm kritiese elemente van die sagteware ontwikkelingsprosesse. Wanneer sekere belangrike elemente van die ontwikkelingsproses nie in ag geneem word nie, loop die sagteware projek die gevaar om te faal. Prestasiemeting tree op as 'n belangrike hulpmiddel vir programmatuur ingenieurswese wat riglyne bied vir die keuse van tegnologiese komponente en projekbestuur wat projek sukses moet verseker.

CHAPTER 1

BACKGROUND, PROBLEM STATEMENT AND HYPOTHESIS

1.1 INTRODUCTORY REMARKS

The history of the Internet and Computer Science has shown that continuous change is inevitable, and it happens at an ever increasing rate. This is obvious, considering the relative short history of Computer Science. Computer Science finds its roots in two distinct fields: Electronic Engineering and Mathematics. Electronic engineering is concerned with the physical operations of a computer, i.e. the transmission of electric currents between the physical components of a computer, while Mathematics focuses on the programming of actions, based on a binary representation of data. A major component of circuit programming is Boolean algebra, whereby logic is built into a circuit for computation. It is solely based on the validity of a statement. 'True' is represented by a one (1), and 'False' by a zero (0). The theoretical work on its computability started in 1930, but the algebra itself was already created in the 19th century. The studies of Boolean algebra reached a great milestone of Computer Science in 1936, with the development of the Turing Machine (Brookshear, 1994:424). It was a conceptual system that manipulated an infinite string of zeros and ones.

These humble beginnings evolved into pragmatism, when the UNIVAC, a programmable computer, was sold for more than one million dollars in 1951. It is considered to be the world's first commercially marketed computer. In 1958, International Business Machines (IBM) decided to add computers to their business product line. FORTRAN was used as the first high-level computer language for application in mathematics, science and engineering in 1957. By 1960, the United States had about 6000 computers in operation, which grew to three times as much in another four years. This was also the same time the first local area network (LAN) was implemented. In 1970, the first 4th generation computers emerged, boasting LSI chips that contained up to 15,000 circuits. Two years later, UNIX was developed using C, and there were already 600 companies operating in the computer market. ARPANET, the predecessor of the Internet, became operational in 1973 and in 1975, the first computer store opened in California and Usenet, a multi-disciplinary network was created in 1979 that hosted news- and discussion groups. There were over 100,000 UNIX installations running world-wide by 1984 and there were

1000 Internet hosts, with 13% of US households owning computers. In 1985, Microsoft released Windows 1.0 and started work on Windows NT in 1988, which is currently a very popular corporate operating system for the PC. This coincided with the release of MS-DOS 4.0. By 1991, 25% of US households owned a computer. Marc Andreessen developed Mosaic, a graphical point-and-click browser program for the Internet in 1993. 1994 saw the launch of the Internet search engine Yahoo.com, as well as the formation of the World Wide Web Consortium (W3C) with over 10,000 Internet sites on the web. In 1996, more than 9 million computers were connected to the Internet that hosted more than 650,000 sites. The year 2000 was predicted as the year of global recession due to a date compatibility problem identified in old computer systems, which lead to worldwide corporate investment of billions of dollars to counter the problem, guaranteeing compatibility when the year 2000 arrived. Also in 2000, one of the worst computer viruses in history infected millions of computers within 6 hours, causing major financial losses to corporations worldwide (Bozdoc, 2000:www.bozdoc.com).

The advancements in the field of Information Technology were foretold by Intel veteran, Dr. Gordon E. Moore with the formulation of Moore's Law in 1965. This law stated that the number of transistors per integrated circuit will double every few years. This "law" still holds after almost 40 years with Intel's "conformance" to his Moore's philosophy (Moore,1965; Intel,2002:www.intel.com) as depicted in Figure 1.1. In 1989 Gelsinger, Gargini, Parker & Yu (1998:47) questioned the possibility of designing and testing a 50 million transistor chip within a reasonable time. Their question is answered by Intel themselves in 2000 by building a chip that houses 42,000,000 transistors (Intel,2002).

This short time-line that highlighted some key events in the history of computer evolution shows that in a matter of a mere 50 years, Computer Science is one of the fastest growing sciences in modern times, based on its global presence in various fields, including finance, education and medicine. Computers also had their impact on commerce, as it is seen on the commercial application of technology and the Internet today.

The Internet started as an academic network, then tested and implemented for military use and has now become a commercial necessity (Leiner, Cerf, Clark, Kahn, Kleinrock, Lynch, Postel, Roberts, Wolff, 2000:www.isoc.org). It is so important to many companies, that should the Internet no longer exist, it could cause a major worldwide recession. The direct effect of information technology in business and the economy is not always clear, however, with the recent decline in international markets, the damage was experienced in the Information Technology field as well. The technology stock market's best indicator, the NASDAQ composite index has fallen from over 5,000 points to just above 2,000 points in a matter of 12 months (Glasner, 2001: www.wirednews.com). This proved to have a severe impact on the United States' economy, that led to the decline of consumer confidence and the subsequent interest rate cuts implemented by the United States Federal Reserve to keep their economy out of a recession (Ulick, 2001:money.cnn.com).

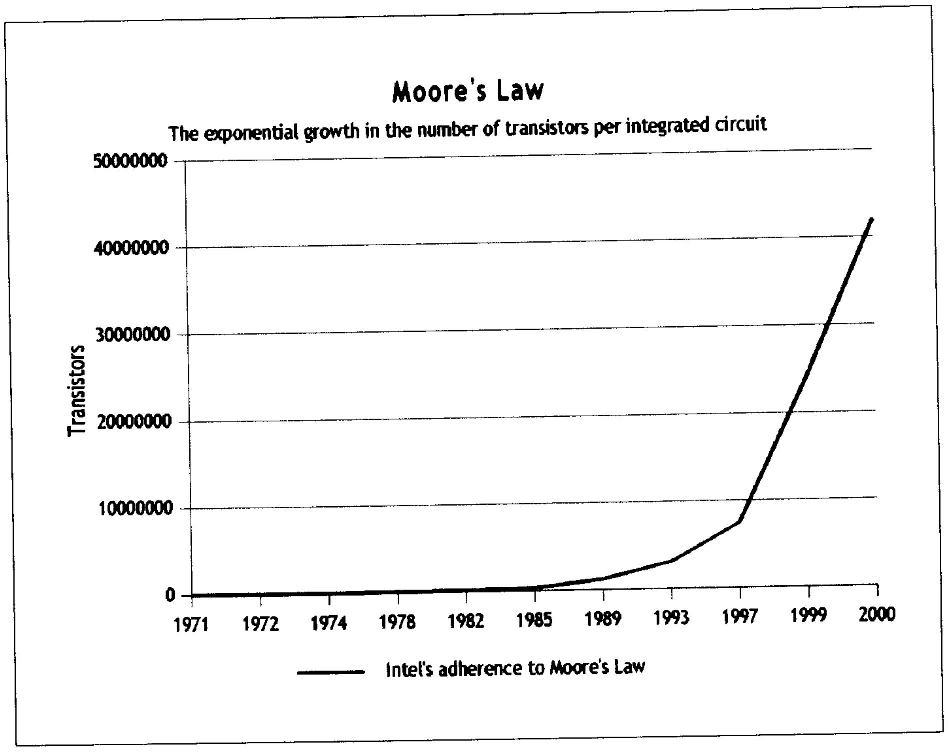


Figure 1.1 Moore's Law

All these factors combined restates the enormous business potential of Information Technology and the Internet. Business is effectively the software developer's client and software engineers and developers have to be able to cater for business's changing requirements. Pressman (1997:836-837) argues that the future of software engineering will be driven by software technology which will force the science into more abstract areas, like artificial intelligence and expert systems. An evolutionary software approach will probably dominate all other development paradigms, reiterating the dynamic environment of this field.

It is clear from the following example given by Willcocks and Sauer, et al (2000:14-17) that software products and technology has to evolve and grow with business to ensure that the best service is delivered to business' clients. Charles Schwab, a United States-based stockbroker, had no Internet exposure prior to 1996. In only four years, this company's client base grew to a staggering 3.7 million online clients, with $800 billion worth of funds under management. It handled more than seventy percent of their 1.7 million trades per week over the Internet with a total value of 25 billion dollars. Accommodating this growth could never have happened without the constant renovation and innovation of technology. Schwab CEO, Charles Schwab reiterates this by making the following statement (Willcocks and Sauer, et al, 2000, 15):

> *"The transforming event is the ability to deliver personalised information to the customer, at virtually no cost...this is possible because the Net is totally imbedded in the centre of our business."*

This success is solely based on technological progress and innovation and provides an example of a perfect merger between business and technology.

1.2 PROBLEM STATEMENT

1.2.1 Background of research problem

Ahituv, Neuman & Riley (1994:328) suggest that whenever a system developer is faced with a new system or problem, he[1] should first review current technologies to evaluate it as a viable technology for implementation. Unfortunately, due to a number of factors this does not always happen. Resistance to change is one of the major factors keeping project teams and members from evaluating current processes and comparing it to different approaches and technologies. By constantly using traditional approaches, certain rewards are still obtained in terms of development speed. Unfortunately, the opposite is also common where serious re-development is required, as the implemented technologies were not the best tools for the application. It can have a serious impact in meeting deadlines throughout the development process, and limit product innovation.

1 For the purpose of uniformity, it was decided to unify reference to people in the male form (his/he)

Many methodologies have been formulated to improve the software development process. One of these processes is re-engineering. Jacobson, Ericsson & Jacobson(1995:15) describes re-engineering to have as aim, the re-evaluation and re-conceptualisation of an existing system and trying to find better and more efficient ways to perform certain actions. Carnegie Mellon's SEI adds that it is to facilitate the disciplined evolution of a system from its current state to a desired state (1995:6). It is further mentioned that re-engineering consists of two distinct sets of activities. Firstly, it supports program understanding and secondly it includes activities geared towards software evolution. Combining several proven methodologies around software engineering can lead to a synergy of results, delivering more than the individual results combined. When re-engineering principles are applied within the software engineering environment, the development process will have the necessary equipment and skills to gain better information on the proper course of action to be taken, taking cognisance of the environment, its technological components and resources required to produce a better product.

It is fairly common to overlook basic factors in the development of new systems. Many developers are used to a certain technology and would continue to use it, rather than expanding their skills, using new and unfamiliar technologies. Ahituv et al (1994:28) argues that resistance to change affects all people involved with the use of a new or unfamiliar system. This problem does not only exist with system users, but also with developers. Some symptoms of this "disease" include:

- The objection to the introduction of new technology;
- Attempts to block the acquisition of equipment by end-users; and
- The resistance to the formation of new organisational units.

These symptoms manifest themselves through several seemingly unrelated side effects. When developers' minds are closed to new energy and ideas, the team and their projects will suffer from it. Stagnated approaches and methods also stifle creativity and can lead to poor innovation. Eventually it can have disastrous effects for the team, its outputs and the clients.

It is common that the professional experiences a feeling that he lost control over his area of expertise and finds himself in unfamiliar territory. Kast and Rosenzweig (1985:220) state the following:

"Technological changes may create job insecurities and anxieties for workers. Skills developed over a long period may become outmoded, vitally affecting their self-image and motivation. We frequently underestimate or fail to perceive the impact of the technical system on people. With stable operations, the interaction goes unnoticed. However, a major change in the technology component will often highlight this interdependence."

The employment of people in technical or scientific areas has shown that their position inside the company is often cause of conflict. This is due to the high value they place on intellectual achievements and innovation. The problem is generally that they are not recognised for their achievements within the company and amongst their professional colleagues. They are in search of a high level of automation, discretion over task assignments, professional recognition and a high sense of personal growth from their work (Kast and Rosenzweig, 1985:223). Management is to play a critical role here as the catalyst that introduces new technology and to facilitate the urge to learn new skills.

Developers and their managers, therefore need to learn to adapt to change by constantly evaluating their methods, tools and environment. Change management is a critical and effective, tool to deal with these problems. Change management has three distinctive phases (Ahituv et al, 1994:29):

- *Unfreezing* Disturbing the current stable equilibrium and, consequently introducing the need for change;
- *Moving* Presenting new directions and conducting a learning process until the material is thoroughly digested; and
- *Re-freezing* Integrating the change with existing behavioural frameworks to recreate a whole, natural entity.

As business is constantly pressured to deliver more at a higher speed, it is important for developers and system engineers to realise a technology's advantages and shortcomings.

In order for developers to deliver high quality software to business, based on its requirements, ongoing technological evaluation and evolution has to be at the heart of all software engineering activities. Apart from quality factors, performance factors should also be evaluated. NetCraft (www.netcraft.com) has developed a discovery script whereby they query several web servers to discover their nature. By the end of September 2002, the statistics they compiled showed web server deployment figures as follows:

Developer	No. of servers in use	Market Share
Apache	21,421,748	59.91%
Microsoft	10,433,095	29.18%
Zeus	742,781	2.08%
iPlanet	485,818	1.36%
Rest	2,672,994	7.48%

Table 1.1 NetCraft Web Server Survey
http://www.netcraft.com/survey

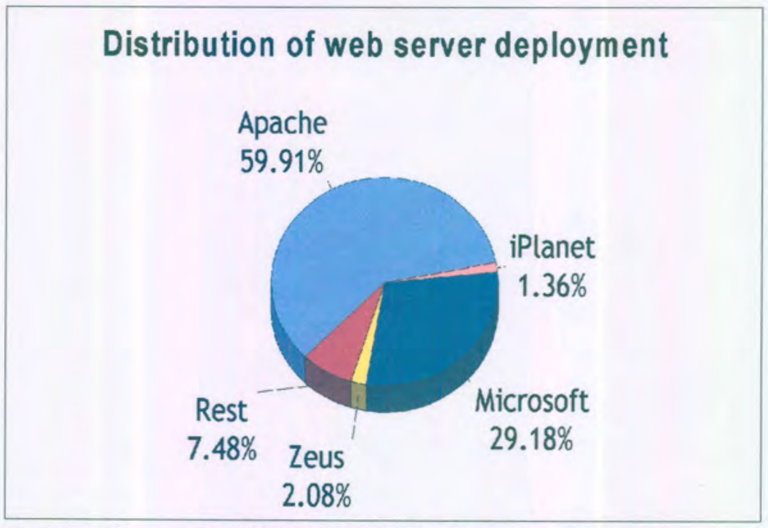


Figure 1.2 Distribution of web server deployment

It is quite clear from this that the Apache Web Server and Microsoft's Internet Information Server are the two most common web servers on the Internet today. As Apache is freely available, it is an attractive option, which might explain its popularity. It is however much more complex to set up and maintain than its Microsoft counterpart. Apache is supported on several platforms, including Linux, UNIX and Microsoft. Internet Information Server (IIS) is bundled with Microsoft Windows in two versions and is only available for Microsoft platforms. The Personal Web Server (PWS) is used for deployment on workstations, and the full server version, for Microsoft Windows NT and 2000 Server. IIS's management is GUI-based (Graphical User Interface) and very easy to install and maintain with a Management Console that allows the administrator to manage several servers from a single program.

1.2.2 Reasons for Server Deployment Distribution

The reasons for the distribution of web servers are unfortunately not clear. Several contradictory reports were found that argue that either Apache or IIS were the better performer of the two.

A troubling report, especially for Linux and Apache, was found on performance tests between the two top servers, done by MindCraft hailing Microsoft's IIS as the undisputed champion of web servers. The report was received with dismay within the Linux community, questioned the fairness and objectivity it was conducted under (Raymond, 1999:www.slashdot.org). Furthermore, Microsoft's Windows platform manager in South Africa, Ian Hatton, admitted that the tests were sponsored by Microsoft (Pienaar, 1999:www.itweb.co.za).

One could argue that the uproar from the Linux community did not have any merit. The Microsoft server could have been faster and the tests were conducted in a proper manner. Why is it then that Apache is more popular than Internet Information Server (IIS)? Is there any correlation between the popularity of the server and its performance? The popularity difference could lie in functionality or convenience. It could even be related to stability.

Should one try to explain the deployment figures (Figure 1.1), one should compare several different variables to determine the reason behind it (MacVittie, 2001):

- Platform;
- Supporting technologies;
- The service application;
- Technical knowledge required to configure and maintain;
- Security requirements;
- Reliability;
- Vendor support and continuous improvement;
- Quality of the API;
- Availability of server extension through plug-ins; and

- Performance.

In general, very few web sites start big. They all had small beginnings, and then, based on their popularity either exploded and became big and successful, or closed down. More often than not, these sites do not have the financial backing like the large corporations like AOL Time Warner's CNN.com, and had to make it on their own. The only way for a small startup site to be seen on the Internet at a reasonable cost is through a shared hosting solution. This means that the site is hosted remotely in a server farm that is closely linked to a major Internet backbone. The costs are reasonably low, and high volumes of data can be serviced by this solution. Based on the owner's financial position, he might opt to go for one of the two most commonly offered technologies for shared web hosting. Either Linux/Apache or Microsoft Windows/Internet Information Server. Usually, these services cost roughly the same, which leaves the owner to decide. By comparing different server technologies and their offerings to his requirements, technology will be the driving force behind these kinds of business decisions (Pressman, 1997:835).

1.2.3 The research problem and hypothesis

To select the appropriate technology for deploying a new web site or service, the owner should always consider a wide range of parameters (paragraph 1.2.2). For a technology to be selected, it should therefore excel over its opponents with a fair margin. One can now even speculate further to say that the server that displays the best set of qualities, when compared to the other, should automatically be the most popular.

To address the research problem, a null- and research hypothesis is formulated. These hypotheses are centred around the correlation between a web server's popularity and its qualities. This study has therefore as goal the proof of the popularity rating as direct measure of a server's overall rating. It is expected that these statistics will show that the more popular a server is, the better would its overall performance be. The hypotheses are formulated as follows:

- **Null-hypothesis** (H_0): There exists no correlation between a web server's popularity and its performance, functionality and reliability;

and

- **Research-hypothesis (H_1):** A web server's performance, functionality or reliability is related to determining its popularity.

In order to prove or reject either the H_0 or H_1 hypothesis, a couple of sub-hypotheses were formulated to combine either an acceptance of rejection of either of the hypotheses mentioned above. These sub-hypotheses are named and discussed in Chapter 3.

1.3 RESEARCH GOALS

As noted in paragraph 1.2.1, when managing resistance to change, it is important to prepare developers with the introduction of new technologies in a cautionary way. By getting them involved in the choice and testing of new technologies, it can greatly improve the introduction thereof. It is however, always difficult to determine by which standard to measure certain characteristics of a system. This research will try and add some insight to the measurement techniques of system components. Furthermore, it might shed some light on using popularity as an effective way of estimating a system's overall success. It should however be stressed here that popularity figures should never be assumed to be the ultimate measurement in software evaluation, but it can add some value in focussing system evaluation exercises. This study will provide a mechanism for more objective decision-making in the selection of technology.

The additional value could be measured in terms of time-costs. By eliminating certain systems from an evaluation exercise might not be totally objective, but it can save the project much time and effort, should the initial field of systems be restricted to highly popular systems. Again, care should be taken with the application of this methodology, as new technologies might not always be very popular, and could even prove to perform better than existing technologies. This reiterates the importance of information technology professionals' interest in new developments in their area. New technological developments should always be considered before the final group of test systems are put through the evaluation.

The inclusion of outside contenders can cause an uneasy surprise. Redhat, one of the well-known Linux distributors in the world has recently launched their own web server. The Tux web server is reviewed as one of the fastest web servers around. This was determined with a benchmark comparison between Apache, Microsoft IIS and the Redhat Tux. Some amazing performance differences were discovered:

> *"Tux's amazing performance benchmarks on the SPECWeb 99 benchmark), found that Tux was able to perform nearly three times faster than current Web server mainstay Apache (12,792 transactions per second vs. 4,602 tps) when running a mix of dynamic and static Web content."*
>
> and
>
> *"The fact that Tux 2.0 was also significantly faster than Windows 2000's Internet Information Server 5.0 Web server (5,137 requests per second) clearly shows the advantages of Tux's new design over that of a well-established Web server."*
>
> (Baltazar, Dyck, 2001:www.zdnet.com)

The great difference in performance is apparently achieved by the fact that Tux's kernel is present within the operating system's kernel, which is different from the other two popular servers, which have their main processes running outside the operating system's kernel. This apparent difference shows that the "vertical position" inside the operating system's architecture is very important. The closer the process runs to the kernel, the higher its performance could be. This has shown so successful, that Microsoft has announced plans to release a similar architecture in their new web servers soon:

> *"The next version of IIS (which ships with Microsoft Corp.'s Whistler project) uses several ideas introduced by Tux, including the kernel-space design."*
>
> (Baltazar, Dyck, 2001:www.zdnet.com)

The exclusion of Tux for this evaluation was based on the fact that it doesn't include certain functions that the other two servers offer. In order to support server extensions and executable binaries, Tux has to be configured to pass requests to an additional Apache server. This functionality forms an integral part of the tests performed later in this study.

When an analysis of this nature is expanded to include several new or unknown technologies, many interesting facts can come to light. Business does not really care about which technologies its Information Technology department comes up with. As long as it can do the work they are satisfied. Add to this the option of a lowcost alternative and you might have a totally different scenario. It is therefore imperative that business introduces more rigid processes to guide and manage decision-making activities. Using several business tools, like re-engineering and risk management, the teams responsible for these processes can keep the company out of danger's way, while ensuring excellence at many levels of the organisation.

1.4 STUDY LAYOUT

This research is presented in five chapters. **Chapter One** introduces the reader to the background of this project, highlighting the history and growth of the Internet, Computer Science and software engineering. The growth of technology has effectively made software developers the information and knowledge agents of business. Furthermore, technology is increasingly becoming the central point of many businesses, emphasising the importance of high quality development and product support. More pressure is placed on developers to deliver bigger and better systems which require the developer to evaluate supporting technologies to ascertain their viability as a supportive system.

Chapter Two is concerned with expanding the foundation with a qualitative study into the systems under evaluation.

Chapter Three focusses on performance issues between two servers. The results are statistically compared and discussed, and provide a good profile of both servers under stress. This profile is used to determine the acceptance or rejection of the null hypothesis stated in Chapter One.

Chapter Four uses the performance results obtained from Chapter 3 to construct a high performance web application. This application was developed for both platforms to give a good comparison of the technologies used for the two different servers. The results from the separate sections will be combined and used to make some recommendations for server development and deployment.

Chapter Five discusses the implications of poor evaluation and control on software development projects. It highlights the importance of risk management throughout the systems development life cycle and suggests areas of continued studies.

1.5 CLOSING REMARKS

Throughout this chapter, several problem areas were identified that could cause a resistance to change with the introduction of new technologies, or issues that could be reason for clouded judgement on the evaluation of technologies. This study has at aim to evaluate popular web servers and comparing the overall evaluation of the servers to their popularity. It is aimed at drawing correlations between their popularity and overall assessment.

Furthermore, it also guides the reader through a basic evaluation technique for evaluating and comparing server technologies. It should be carefully noted that the choice and introduction of new technologies are to be made through a combined exercise between developers and management.

CHAPTER 2

RESEARCH FRAMEWORK

2.1 INTRODUCTORY REMARKS

Chapter 1 discussed the dynamic nature of Information Technology and Computer Science. The fields of Computer Science and software engineering share a common area of interest in the application of existing technologies, the improvement of insufficient or obsolete ones, and the development of new technologies. Fenton and Pfleeger (1997:9) state the properties of software engineering to include management, costing, planning, modelling, analysis, specification, design, implementation, testing and maintenance.

The evaluation of technologies is therefore of the highest importance to this field of study. Evaluation and measurement does however seem to be of lesser importance currently in the area of software engineering (Fenton & Pfleeger, 1997:10). This highlights a serious problem with the modern approach to software engineering. Measurement is considered a mere luxury for software developers and project managers. When measurement is applied, however, it tends to be infrequent, inconsistent and incomplete. This is clearly not conducive to overall project success. Four critical factors involving poor software measurement are highlighted (Fenton & Pfleeger, 1997:10):

- Failure to set measurable targets for software products. Often "fuzzy" objectives are set that are hard to measure at a project's completion;
- Failure to distinguish between the component costs of the project. For example, most projects cannot differentiate the cost of design, coding and testing. Cost control is difficult with poor cost assignment;
- Failure to quantify and predict the quality of the product. This has the effect that the finished system cannot be benchmarked against estimated performance figures or system specifications; and
- Often new and revolutionary methods and products lure the developer into using it without scientific analysis or proof why it should be used and old technologies are to be discarded.

From an objective point of view, it seems unthinkable that software development projects do not avoid the pitfalls mentioned above. However, these pitfalls are real in the world of software development and therefore emphasise the need for exact software measurement.

These sentiments are echoed by Gilb's Principle of Fuzzy Targets (Fenton & Pfleeger, 1997:10):

"Projects without clear goals will not achieve their goals clearly".

2.2 THE NEED FOR SOFTWARE MEASUREMENT

As stated earlier, measurement is a critical tool by which we compare success, especially in the areas of engineering. This technique applies to software engineering as well. The ultimate goal of software measurement is the realisation of pre-project specifications and requirements. It also helps in assessing the overall health of a project. Good measurement and management leads to effective control (Fenton & Pfleeger, 1997:11). Furthermore, decisions can now be made, based on concrete measurement-based information. This is even more true for the evaluation of new technologies. In order to effectively evaluate supporting technologies, the developer has to measure the different characteristics of different technologies. Unbiassed evaluation is required from the programmer and his team. Disregard of standardised testing may result in skewed outcomes.

The value of measurement in the evaluation of technologies is therefore summarised as follows:

- It will ensure the objective comparison of different technologies;
- It will identify both good and poor characteristics of systems;
- It will provide confidence to the project team that the technology applied will operate within a set range of parameters;
- It forces the team to compile a thorough list of requirements for the system or technology, which will be useful in the ongoing evaluation of the technology and will provide a guideline by which requirements will be met with the project's completion.

2.3 MEASUREMENT FRAMEWORK FOR EVALUATED SYSTEMS

In order to structure this research properly, the tests will be grouped in two distinct sections. Firstly, the qualitative section will look at the soft, non-quantifiable characteristics of the systems involved. This section will cover several issues, including installation procedures, aesthetics and general management and maintenance. The second section (Chapter 3) of the research will focus on the quantitative aspects of the different systems evaluated. This entails the comparison of performance figures and the like.

2.3.1 Microsoft Internet Information Server (IIS)

Microsoft was a late joiner to the web service party, but having said that, they have made quite a noticeable impact on the web server market, as depicted in Table 1.1. Even though earlier versions of Microsoft's web server were clearly not as nimble and gracious as their fifth, and current, version, Microsoft's IIS was so popular over the last few years that it is currently the second-most common web server on the Internet (NetCraft, 2002). The server is not a core component of Microsoft's operating systems, but is however, dependent on it. IIS is not platform independent, which means that you need a Microsoft operating system to deploy IIS. This can be either Microsoft's Windows 9x series, Windows NT, Windows 2000 or the latest, Windows XP. It should therefore be said that Microsoft provided for some kind of flexibility with regards to IIS, even if it limits the user to a single Operating System vendor.

There are different versions of IIS. The basic desktop version, Personal Web Server, is intended to be deployed on a workstation-style operating system. This version supports basic Hypertext Transfer Protocol (HTTP) and File Transfer Protocol (FTP), but in a very limited fashion. This version is much less versatile than the server version. The server version is capable of hosting multiple sites, which makes it an ideal platform for shared hosting. This means that one server can be used to host multiple sites, which means lower cost of ownership and simpler management.

Unlike IIS's rival Apache, IIS is proprietary software. The source code belongs to Microsoft and only them. No-one else knows the inner workings of IIS and Microsoft intends to keep it that way. This has further implications for the owner. With IIS being a black-box[1] installation, it limits the administrator to be able to configure the server within a maximum parameter set that Microsoft allows. This does however offer an advantage for novice administrators. This is a plug-and-play web server that does not require any difficult configuration from the administrator. Furthermore, Microsoft has proven over time that they are able to deliver a system that is already configured to deliver a high performance solution to the system owner.

2.3.1.1 Installing Internet Information Server

Microsoft is a master in building and constructing good-looking graphical user interfaces(GUI). This is true for all their products, from installation, through operation up to software removal. This goes for the IIS installation as well. IIS 4.0 gives the installer full control over installation. It allowed for the selection of both installation paths, and component installation. Once all required options and interactions were executed, the rest of the installation continues unhindered with the file copying and system configuration.

At the completion of the installation, the server is operational. The service is automatically added to the operating system's list of services started at boot-time. A restart is recommended and after the system has booted, IIS is running. This easy installation was duplicated in IIS 5, which comes bundled with Microsoft's Windows 2000 Server Family. It is optional with the setup of the operating system and can be added or removed at any stage.

By default, IIS will create a default web site on port 80 (the default HTTP port) on the machine's main IP address. IIS also includes an optional FTP server. A default FTP server will also be created together with the HTTP server. As part of the default HTTP server, the installation also installs the required documentation which can be accessed as part of the default web site.

[1] 'Black-box installation' refers to the closed nature of software and its internal processes.

2.3.1.2 Maintenance and configuration

Internet Information Server can be managed in two ways. Firstly, the most commonly used is the application-based management console. It gives the administrator full control over all the virtual servers set up on a machine. The management console has the ability to log on to a remote server and allows the administrator remote administration access to any other machine running IIS. The console is snap-in driven and forms the basis of several management modules to be installed and run from there. This concept is similar to the console Microsoft distributes with their flagship database server, SQL Server. The database is managed and maintained through the console, similar to the IIS server.

Secondly, like many other servers, IIS can be managed through a web-based console that gives the administrator remote administration functionality through any modern web browser. Though not as powerful as the application-based system, the web-based console is quite powerful in general maintenance of the server. These two different maintenance ports offer therefore an easy configuration framework of the server in a GUI-based environment. High level optimisation and performance tweaking can however not be done in any of these modes. In order to obtain full efficiency, minor adjustments to the server have to be performed on Windows' registry. This is not recommended to inexperienced administrators and requires some knowledge of the operating system's registry.

Microsoft's IIS supports multiple virtual servers. A virtual server is a virtual computer that resides on one physical computer but appears to the user as a totally separate server. Each virtual server on the machine has either its own IP address or port and can service its own domain. This allows a company to host several different web sites on one physical machine at a relatively low cost. Microsoft's range of server products allows for multiple IP addresses to be assigned to a network interface card (NIC). All the addresses that are assigned to the NIC can be used by IIS to create a virtual server on one of 2^{16} ports. This functionality is however only available on the high-level server edition of IIS. This is due to the underlying operating system's shortcomings with regards to multiple IP address assignment. It could be argued that it was limited due to commercial motivations. By limiting the functionality to host multiple sites on Microsoft's server products only, it created a huge income from corporate clients and hosting companies.

IIS also supports virtual directories. A virtual directory allows the administrator to link directories outside the server's root directory to be accessible as a normal subdirectory within the root. It holds major advantages in terms of resource management and control.

IIS can run as an out-the-box solution. Just install and run. Through their marketing campaigns, Microsoft claims furthermore that their new generation servers proved to have an uptime figure of 99.999% (Microsoft, 2001). This is a good indication of IIS and Microsoft's reliability and stability. It should however be kept in mind that actual industrial deployment might have different outcomes to that of development-testing scenarios.

2.3.1.3 Optimising Internet Information Server

Basic optimisation can be done from either of the two management consoles. However, to really get IIS to perform well, some optimisation has to be performed at the operating system's registry level. This is a very basic configuration framework, but one small change to the wrong value will slow down the server and can even affect the operating system's stability. Fortunately, there is sufficient documentation available from Microsoft on optimising IIS.

Trying to optimise this level of IIS should only be attempted once the basic optimisation fails to deliver the required results.

2.3.1.4 Server Architecture

Like many other web servers, Microsoft's IIS web server defines a basic functionality that can be used to build web applications. Active Server Pages (ASP), Internet Server Application Programmer Interface (ISAPI) and other technologies extend the basic IIS web server to allow developers the implementation of technology to develop high-level programs that can run on top of a web server. This allows the developer to build in functionality that are not available with straight-forward HTML, such as mathematics, logic, data streaming and database access. The core functionality of the IIS includes:

- Establishing and maintaining HTTP connections;

- Reading HTTP requests and writing HTTP responses;
- Modifying HTTP headers;
- Obtaining client certificate information;
- Managing asynchronous connections;
- Mapping Uniform Resource Locators (URL's) to physical paths;
- Managing and running applications; and
- Transmitting files.

The architecture that Microsoft applied with their IIS is shown in Figure 2.1. This representation of the Microsoft web server shows that the server has a core Transmission Control Protocol (TCP) server engine that handles basic HTTP requests to HTML, images and other files. This engine is expanded to allow several other technologies to interface with this low level server, allowing the user to customise the actions of the server to provide much more complicated and involved functionality (Microsoft, 1999):

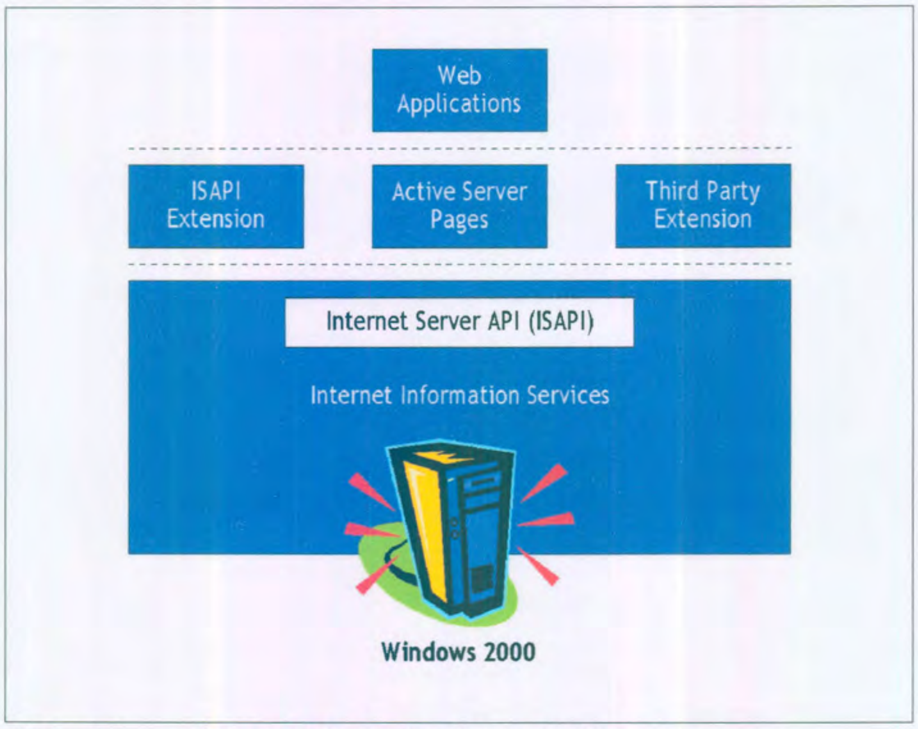


Figure 2.1 Internet Information Server Architecture (Microsoft, 1999)

IIS supports a fully threaded model to handle multiple requests. This functionality is extended throughout the server, including plug-in management. IIS's ASP support is also capable of performing self-tuning through thread management. Inside the server, threads are maintained in a thread pool, that is capable of autonomous management to improve the server's overall performance.

IIS distinguishes between its handling of HTTP requests. Four different processing options are noted by Microsoft:

1) **Static HTML Pages**
   HTML (also XML, CSS, etc.) documents are served by the server to connecting clients. These documents are static in nature and is served to the client without any further pre-processing on them.
2) **ISAPI Extensions**
   IIS loads the ISAPI DLL and the request is sent to the extension through the **Extension_Control_Block** data structure. IIS allows for DLL caching. This drastically improves performance, as the DLL is loaded only the first time the DLL is accessed. Thereafter, the cached extension is accessed at memory speed, compared to disk-read access-speeds.
3) **File name extensions mapped to a particular ISAPI extension**
   As mentioned previously (see 2), the DLL is loaded and IIS presents the request to the extension. The ASP extension, for example, is mapped to the asp.dll extension, and all requests to these files are handled by the appropriate DLL. This forms the basis of ASP operation. Requests to the document will be passed to the mapping library, where it will load the document requested and perform pre-processing on the document and send the output to the requesting client.
4) **CGI applications**
   Different from the ISAPI requests, with CGI, IIS has to create a new process whereby the CGI-thread is handled on its own. IIS will provide the query string to the CGI-thread process. By definition this method of execution should be much slower than ISAPI.

2.3.1.5 <u>Access logging</u>

Internet Information Server has its own internal high-speed logging system that can be used

i16379354
b15821663

with the server. The logging modules operate independently from the core processes on the server and offers the user a range of logging formats. The logs are compiled separately for all virtual HTTP- and FTP servers. Logging is usually done to text files, but IIS also provides for ODBC-logging (Open Database Connectivity). This enables the administrator to have the logs recorded straight into a relational database for SQL-based manipulation. Apart from the ODBC logging format, IIS supports three other log standards, being the W3C Extended Log File Format, Microsoft IIS Log Format and NCSA Common Log File Format.

Logging is not compulsory, and can be disabled. The main reason to do this is to minimize system overhead within the server, to obtain better system performance. Different logging formats and preferences can be set for all virtual servers (both HTTP and FTP) from the different management consoles. All server activities are logged by the server.

2.3.1.6 <u>Security</u>

IIS supports both anonymous and user-authorised access. Anonymous access is the most common access-level set for all web sites. Where access to a server or segment needs to be restricted, user names and passwords are required. IIS supports Distributed Authoring and Versioning (WebDAV) for exposing any storage media over an HTTP connection. WebDAV is configured by using web server permission settings. This implies therefore that the user logs on to the server as a remote user. WebDAV permissions can be set to allow the following (Microsoft, 1999):

- File- and directory search;
- Creation, modification, deletion and browsing of directories and files and their properties;
- Storing and retrieving custom properties for files and directories; and
- Locking files for collaborative working environments.

IIS's access control mechanism covers four basic access criteria:

- IP address;
- User permission;
- Web server permissions; and
- Filesystem permissions.

Even though Microsoft has claimed many times that the security they build into their servers are of the highest standard, their systems seem to be compromised quite often. These problems are so big that should a server's security be compromised, and the attacker is familiar with the structure and location of the server, he can cause serious damage to the server and the network it is running on. This has been emphasised by the high number of security-related updates released by Microsoft for their servers. Ritchley & Hamilton compare the two servers' (Microsoft IIS and Apache) vulnerability histories in terms of severity and exposure time and found that Apache is in terms of security superior to IIS (2001).

Based on these issues and many others, many people argue that Microsoft's attention to security leaves much to be desired. Others argue that Microsoft is only the target of bad publicity with the discovery of security holes. Abreu (1999:www.cnn.com) quoted Jeff Tarter, editor and publisher of Softletter, saying that Microsoft's software was never really invented to operate in a networked architecture. As Microsoft and the rest of the software world is moving to a complete network-oriented architecture, this problem can prove to be fatal to their future successes. Furthermore, the implications for multinational companies, like DaimlerChrysler, Deutsche Bank and Ford which rely on these technologies, can be to their detriment.

Bruce Schneier, cryptology expert, mentioned that Microsoft's operating system was never developed with security in mind (Abreu,1999:www.cnn.com). Furthermore, he said that for them (Microsoft), security is always an afterthought. Many of these security problems have surfaced in recent times, from e-mail viruses to denial of service attacks from IIS-driven web servers. The latest of these vulnerabilities, at the time of this project, are the malicious "Code Red" and "Code Blue" viruses. This and many of its kind have caused havoc for site owners and administrators over the last few months, just as the dust settled over the Y2K bug and the large investments made by Microsoft to update their legacy software systems.

2.3.2 Apache

Apache is a web server built on open-source collaboration. It has at aim the creation of a robust, commercial-standard and feature-filled web server which is freely available. The source code is open to use and deploy by anyone, royalty-free. The project is jointly managed by a group of volunteers world-wide, which use the Internet as their means of communication, planning and development of the server and its documentation.

Rob McCool developed an HTTP daemon, which was the most popular server software on the Internet by 1995. Development of this public domain server had stalled and many developers had to distribute a number of bug-fixes for the system. A small group of these web masters gathered and formed the original Apache Group, primarily through e-mail communication. They were to coordinate the changes made and oversee the distribution of these software "patches". During the period May 1995 to June 1995, they implemented new features on the National Centre for Supercomputing Applications (NCSA) 1.3 httpd base system. A new server architecture was developed, and after extensive beta testing, a new set of documentation and a myriad of new features and modules, Apache 1.0 was released in December 1995. Apache managed to pass the NCSA httpd daemon in less than a year after inception as the most popular web server on the Internet (Apache Project, 1999:www.apache.org).

2.3.2.1 Installing Apache

Apache is open-source. This means that the webmaster can control everything inside Apache. He can determine the installation directories, its individual configurations, and much more. Apache is available for installation in two different formats. Firstly, Apache is distributed in a Redhat Package Manager (RPM) file format. This allows the administrator to install the server from one installation file to its default location and do not have any control over the finer options in the installation process. This format is only suitable for Linux machines that support RPM. The Apache Group released a new Microsoft Windows-based version of their server, which is also installed by a single installation application, allowing the user to make a few installation configuration changes.

Apache is probably the widest deployed web server in the world today. This can be attributed to its multi-platform support approach. Apache is available for almost all variants of Linux, Unix and Microsoft platforms.

The more involved approach of installing Apache is the compilation process, by which the source is compiled, built and installed. This is the most common for experienced system administrators, as they would prefer to customise certain aspects of the server.

Once you get tired of Apache on your server, removal can be done in a few simple steps, provided it was installed through the RPM (RedHat Package Manager) or any similar installation wizard. However, should you have used the compile-approach of software installation, it can be much more involved.

To enable some advanced features for Apache, like Dynamic Shared Object-support (DSO), the basic installation will not suffice. Certain configuration changes have to be made prior to installation through the compilation-oriented approach.

2.3.2.2 Maintenance and Configuration

Apache is in essence a UNIX-oriented server which means that, like most parts of UNIX and Linux, configuration and management of the server happen through text-based configuration files on the server. The server uses several configuration files, that determine the server's characteristics, ranging from domain resolution to IP- and port configuration and dynamic shared object binding.

For the novice, configuring an Apache server with no previous web server experience might be a nightmare. There are no ready-made management utilities available like the console that is included with Microsoft's IIS. There are a few simple utilities that are installed with recent versions of RedHat, but they do not provide the administrator with the same power as the IIS console does.

Server startup and shutdown are done mainly from a command-line level. Management of the server through remote Telnet sessions is quite simple and extremely convenient, particularly when servers are not located where its administrator is.

2.3.2.3 Optimising Apache

Apache is configured through a series of configuration files located in various directories of the server installation. Needless to say, optimising the server will happen in exactly the same way. A real powerful optimisation option to go for would be to pre-configure the server at installation time. Several documents available to the Apache community recommend the recompilation of the underlying operating system, primarily focussed on the web server's application and performance. However, due to the focus of this research, recompiling the underlying operating system was not considered.

There is a collection of web sites on the Internet dedicated to optimising Apache for maximum performance. This is in line with the Apache Group's dedication to open-standards development of an open-source web server. Server optimisation remains a black art, and this is even more so for Apache. Apache optimisation can run down to the operating system level, which can prove to be really complex and intricate.

2.3.2.4 Server Architecture

Apache is a modular web server. This means that the server is built from several components which perform separate tasks, as opposed to a monolithic server with a single unit that handles all actions. Apache has a core module that defines and coordinate the steps in servicing a request. The different modules of the Apache server implement the actual phases in the handling of the request (See Figure 2.2).

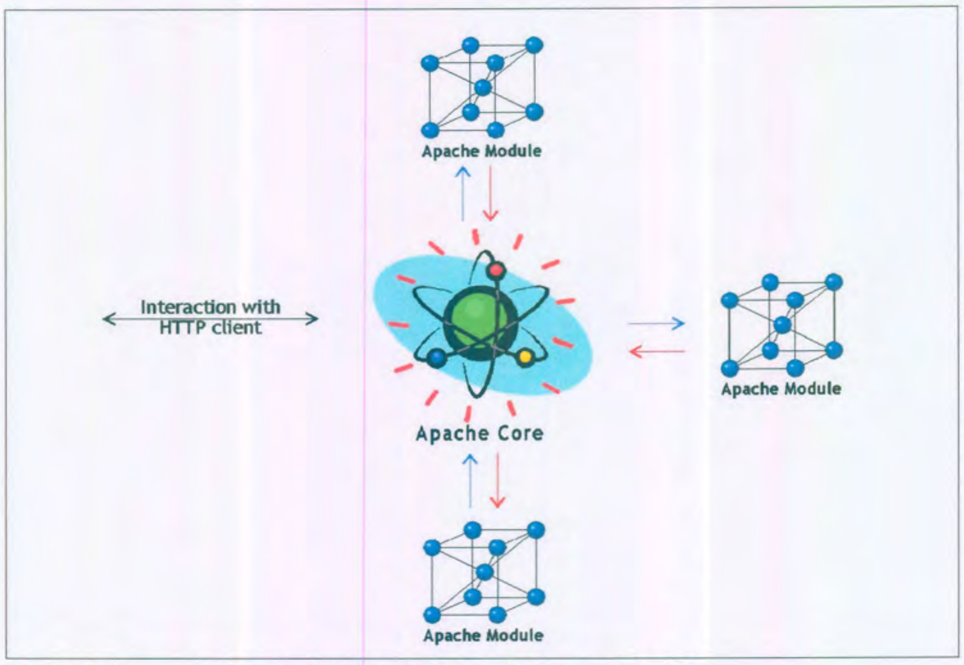


Figure 2.2 Apache high level conceptual architecture

The Apache core implements the basic server functionality with a couple of other utility functions. One of these is resource allocation to the core and server modules. The server's core has the following responsibilities:

- Communicates directly with the HTTP protocol, managing all data transfers;
- Server management;
- Handles errors and the flow of request processing and dispatching control to the modules in the appropriate order;
- Resource allocation and management; and
- Other utilities responsible for reading and managing configuration files and the management of virtual hosts.

Apache's modules are totally independent. They do not communicate with each other but have an interface directly into the core. Should these modules be required to communicate, it will be regulated through the core. Modules implement, override, and extend the basic functionality of the server. Apache allows for dynamic loading of modules as they are required. This allows for better resource management and performance optimisation.

Every new request that comes into a TCP server is assigned to a newly forked child process. Apache applies a technique, called *persistent server processes*. This method implies that the server spawns a pre-determined set of children processes, operating independently in their own address space. These processes will handle incoming connections when required. The Microsoft Windows version of the Apache server spawns a fixed number of threads to handle the load. The different applications of load handling are related to differences in the operating systems. Figure 2.3 shows diagrammatically the way in which Apache applies concurrency.

While discussing processes, it is important not to confuse processes with modules. Modules operate as sub-components of the server. These child processes are separate processes that function totally autonomous with interaction with the connecting client. They do not need to channel communication through the core, or any other central processing module. They communicate directly with the client.

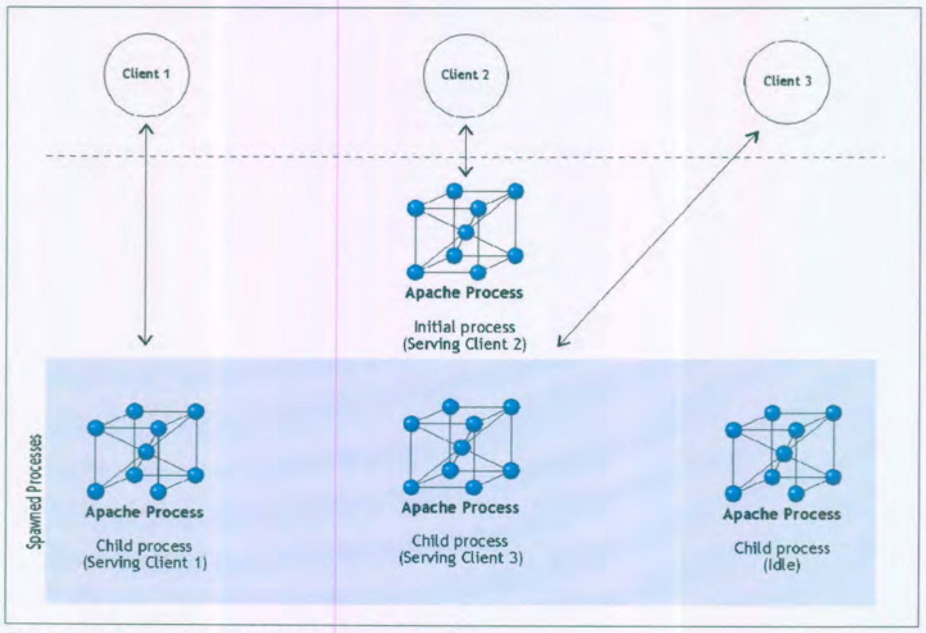


Figure 2.3 Concurrency on Apache

One interesting aspect of Apache's concurrency management, is that it can dynamically control the number of children spawned. This is determined by the current workload placed on the server. The Apache server approximates the *implicit invocation* architectural style (Dragoi, 1999:www.grad.math.uwaterloo.ca). This model centralises control and maintenance in the core module, and is event-driven. Unfortunately, this model can have a performance impact as the event tracking and management of the system can be resource intensive.

2.3.2.5 Access Logging

Every document requested from the server is logged by Apache, as well as any errors that are experienced or missing documents requested. Apache produces log files in the Common Logfile Format (CLF) into flat text files. All basic information is stored, except for the user agent, the referring Uniform Resource Locator(URL) and the use of any cookies. The access and error logs are usually stored in the same subdirectory.

The information that is stored in the access log file includes the following:

- The domain name of the requesting machine. Should the domain not be available through a reverse DNS-lookup, the IP address of the client is used;
- The user name is recorded, should access to the resource require authorisation;
- The date and time on which the request was made;
- The complete first line of the request in quotes; and
- The total number of bytes transferred.

Apache's error logging system is extensive and logs both errors and warnings. Should basic problems occur on the server or its network, it is logged, e.g. when a client times out, it is logged, warning the administrator of possible network problems. Broken links or missing resources are also logged, ensuring the administrator is aware of certain problems with the site. Incorrect authorisation is also logged, warning the administrator of attempted break-ins, or the use of invalid or inactive user credentials (Behlendorf, Chandler, Brintle, Casselberry, Anthony, Darnell, Estabrook, Graber, Hubbard, Ladd, Parker, Scott, 1997:Chapter 16).

2.3.2.6 Security

Apache is synonymous with UNIX and Linux, both of which are highly acclaimed for their security. By default Apache allows access to all resources defined as part of the site to all incoming users. Hontañón (2000:www.networkmagazine.com) questioned Apache's security quality with reference to the fact that it is free. This is however highly debatable, as UNIX is a popular operating system for corporate firewalls. And as the foundation of Linux is ultimately UNIX-based, one can expect similar standards from the UNIX derivative.

Apache supports access control to individual users, based on certain criteria, being domain-specific or IP-address or -range specific. Access to resources can be explicitly allowed or denied. This gives the administrator total flexibility in controlling access to the server. User authentication is also supported by Apache, requiring the user to log onto the server with a user name and password pair. The most common access-mode for web users is however the anonymous mode, which allows all users access to the entire web site. Figure 2.4 shows the security control modules provided with Apache.

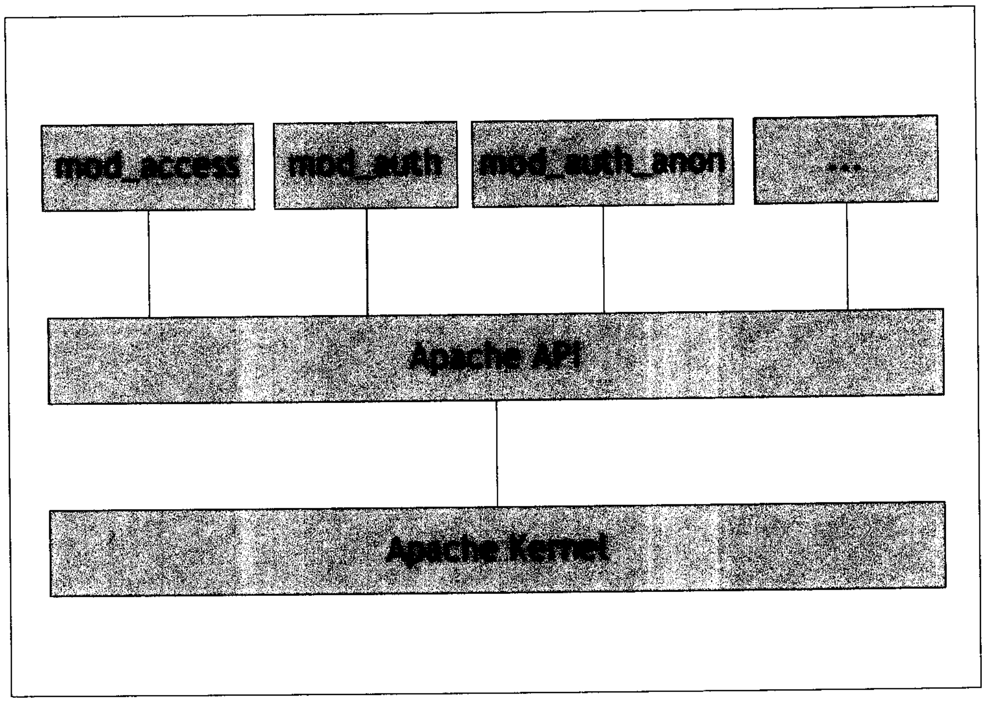


Figure 2.4 Apache Security Modules

The nature of the Apache web server provides the community with well-documented updates to the system, fixing potential security problems.

With the commercial growth of the Internet and the high need for online commerce, it became critical to protect sensitive information exchanged between customers and online retailers, such as credit card numbers, passwords and medical records. This necessitated the creation of the Secure Sockets Layer (SSL) networking protocol. The SSL protocol encrypts data sent through it, guaranteeing total confidentiality.

Apache has developed a Secure Sockets Layer-enabled (SSL) version, Apache-SSL. It is based on the common open source version and supports 128-bit encryption for both commercial and non-commercial use. There is furthermore, also mod_ssl which is a security module that plugs into the standard Apache server. It is the web interface to the popular OpenSSL implementation. SSL is implemented as an extension to the standard Apache source tree. It is also free to distribute for commercial and non-commercial application.

2.4 CLOSING REMARKS

The need for software evaluation was clearly defined and discussed in this chapter. Due to the subjective nature of qualitative evaluation, it was decided that the recommendations made regarding the technologies exclude the qualitative findings for making recommendations. This study will therefore focus primarily on performance measurement from the context of Internet development. It is however quite important to realise that performance measurement is a critical part of all kinds of software development projects.

By combining quantitative analysis with quantitative testing, a project team is able to make objective decisions about technologies. Objectivity and innovation are stimulated by investigation and experimentation, and by combining different methodologies to the entire software development process, better products can be produced, taking cognisance of current trends in terms of old, current and new technologies. Chapter 3 will expand the analysis of this study and look at the quantifiable components of the different technologies being evaluated.

CHAPTER 3
QUANTITATIVE ANALYSIS

3.1 INTRODUCTORY REMARKS

Chapter 2 focussed on the qualitative aspects of the two Internet servers under evaluation, namely Internet Information Server and Apache. Chapter 3 focusses on the quantitative aspects of theses servers, particularly their respective performance under different circumstances.

3.2 TESTING ENVIRONMENT

For the purpose of this study, a stress simulator was built. It consists of one central controller and several simulator clients (satellites). Each of these clients runs on separate machines connected to the controller via TCP. A script is compiled on the server of each of the requests that will be made to the web server. A test is started from the server which sends the test information to the clients, which in turn create the test threads connecting to the web server. As soon as all the clients' threads are created and prepared, the server sends an initialisation command to the clients, starting the suspended threads. A non-blocking HTTP client was used to ensure that the maximum throughput can be achieved for the clients.

The load is distributed evenly over the number of clients. For this test three machines were used. The web server was configured with a 1.6 GHz Intel Pentium 4 processor with a 400MHz system bus. The server used 256MB DDR RAM, and two identical 20GB hard disk drives. The two operating systems, Windows NT 4 SP6 and RedHat Linux 7.2, were installed on separate drives. Both web servers had an identical hardware specification, creating an unbiased hardware platform. Windows NT was configured with Internet Information Server 4 and RedHat Linux with Apache version 1.3.24. The three clients and the web server were connected through a 3COM switch and were all running on a 100Mb full-duplex network configuration.

To simulate various number of stress levels, 32 tests were run for every testing scenario. Each of these tests was run at a higher stress level than the previous one. Starting at 24 threads in total, the stress level was divided over the three clients, starting with 8 threads each. With each iteration, the stress level for each client was increased with 8 threads, up to a total of 256 threads per client, bringing the total stress level to 768 threads. The tests were run at one minute intervals with a 15-second warmup session for each test.

At the completion of every test, the total for each client was calculated and sent back to the central server. The totals were then calculated and logged.

3.3 STATISTICAL ANALYSIS OF DATA

3.3.1 Test breakdown

The tests are structured by means of two different variables, namely the nature of the request and its size. The test can therefore be split into two different sections, being static and dynamic content. Both sections were run with three different sizes of data files[1]. The file sizes used for these tests were not based on any scientific calculations, but on log analysis of several web sites over a period of 1 month. An approximation of the test sizes was made, based on observed clustering within the file size range.

The dynamic section of the test is further divided into three different subsections. As Apache and IIS support similar technologies, the shared methodologies are compared. These technologies include:

- Static files;
- CGI executables; and
- Dynamic Linked Libraries (IIS) and Dynamic Shared Objects (Apache).

The third item (DLL & DSO) is further sectioned into two different subsections. One being cached document retrieval, and file-based document retrieval. Figure 3.1 shows a

1 The three different file sizes were as follows:

Small:	15KB
Medium:	100KB
Large:	500KB

diagrammatical representation thereof.

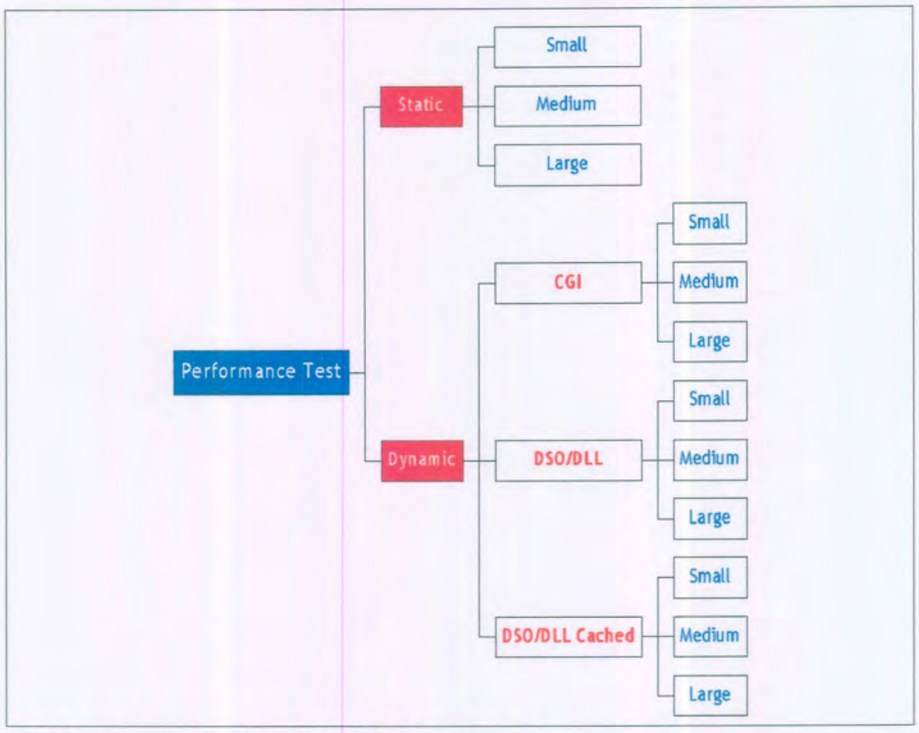


Figure 3.1 Performance Framework

A structure like this provides a good overview of a server's overall performance, as it covers all the major sections of the server's activities. It covers basic document retrieval for small files and the retrieval of documents from an archiving system, including the compilation and manipulation of dynamic documents and graphics. As the tests are run on three different file sizes in both static and dynamic environments, it is representative of the server's activities when run over various stress levels.

3.3.2 **Analysis**

3.3.2.1 Hypothesis testing

When considering two or more samples, it is hard to reach any objective conclusion of the data without statistical analysis. In order to make meaningful deductions from data, statisticians have developed scientific methods for comparing two or more samples. **Hypothesis testing** is a technique whereby decisions can be made from an observed sample based on the correctness of a hypothesis (Steyn, Smit, Du Toit, Strasheim, 1994:407). Hypothesis testing is based on a simple process of stating a hypothesis (making a statement) about the expected characteristics of a sample. Statistical hypothesis testing usually involves the stating of a ***null hypothesis*** (H_0), and an ***alternative hypothesis*** (H_1). The null hypothesis is assumed to be true before testing the hypotheses. Should there be found that the alternative hypothesis holds true, the null hypothesis is rejected and the alternative hypothesis is accepted. Testing hypotheses always run the risk of being wrong. An accepted or rejected null hypothesis will have a probable value of trust associated. A hypothesis with an α-value of 0.05 has a reliability of 95%, or .95 for a one-tailed test. This means that we can say with a 95% certainty that the test yields the correct deduction.

To prove a hypothesis, some underlying tests are required to prove that the data conforms to a predetermined distribution. The statistic calculated for such a test should therefore fall with a reasonable certainty inside the distribution. Examples of these tests include the t test and χ^2 test (Hamburg & Young 1993:309-321).

3.3.2.2 The paired-samples t test

The results of the tests can not be interpreted as independent, as they have a common variable, being the server load as the active threads. Should these two data sets be compared with a normal t test, it will violate the independence condition (Hamburg & Young 1993:358). The test results should be paired by grouping the two servers' performance figures by threads, size and style. The samples are related to one-another and should therefore be seen as paired samples. Two samples of equal size is dependent on one-another when all cases in the one sample can be directly paired with a corresponding point in the other (Steyn, Smit, Du Toit, Strasheim 1994: 435).

Furthermore, according to the model presented by Steyn, et al (1994:452), should a paired-samples t test be applied due to the following reasons:

- Comparison of averages;
- Two samples; and
- Paired values.

The **paired-samples t test** requires the assumption that the underlying population originates from a normal distribution (Hamburg & Young 1993:648; Steyn et al 1994:453).

3.3.2.3 The Wilcoxon matched-pairs signed rank test

Unlike the parametric paired-samples t test, the **Wilcoxon matched-pairs signed rank test** is nonparametric, and it does not make any assumption about the population distribution. It is the nonparametric equivalent of the paired-samples t test (Hamburg & Young 1993:647-648). The Wilcoxon test is carried out by calculating and tabulating the difference between the paired observations ($d = X_2 - X_1$). The absolute values ($|d|$) are obtained, pooled and ranked from 1 to n. A positive and negative rank is compared and is used as the basis for the H_0 hypothesis: $H_0 : \Sigma \text{ rank } (+) = \Sigma \text{ rank } (-)$.

The smaller of the two rank totals are called the Wilcoxon T statistic and used for determining the outcome of the hypothesis test. Appendix 1 tabulates the Critical values of T for the Wilcoxon matched-pairs signed rank test at various levels of probability. The test determines that the null hypothesis would be rejected at the significance level (α) at $T \leq T_\alpha$ (Hamburg & Young 1993:648-649).

Huysamen (1983:179-180) and Hamburg & Young (1993:649) states however that, in cases where n is large ($n > 25$), T is approximately normally distributed, with the following mean and standard deviation:

$$\mu T = \frac{n(n+1)}{4}$$

$$\sigma T = \sqrt{\frac{n(n+1)(2n+1)}{24}}$$

3.3.2.4 The Kolmogorov-Smirnov test for normality

When the normality of the population cannot be accepted per se, a test for normality must be performed on the observed sample. The **Kolmogorov-Smirnov** test compares the observed cumulative distribution function for a variable with a specified theoretical distribution, which may be normal, uniform or Poisson. This goodness-of-fit test tests whether the observations could reasonably have come from the specified distribution (SPSS, 1996:242).

Steyn et al notes the formula for the T and Z distributions as follows (1994:365):

$$Z = \frac{\bar{x} - \mu}{\sigma / \sqrt{n}}$$ for a n(0,1) distribution; and

$$T = \frac{\bar{x} - \mu}{s / \sqrt{n}}$$ for a t(-1) distribution.

Seeing that S will not differ much for an observed random sample from σ, it can be expected that the distribution functions of Z and T will plot roughly the same area, especially with large values of *n* (Steyn et al, 1994:365). The test for normality is therefore done by calculating the Kolmogorov-Smirnov Z. The null hypothesis that the observed sample is sourced from the predetermined distribution is to be rejected if the calculated ratio is greater than the tabulated value at the level of confidence.

3.3.2.5 Application of the test results

By evaluating the different sections of the overall performance test, much information is obtained about the servers' strengths and weaknesses. When these results are later combined, a performance profile can be compiled for each of the two servers. The application of statistical methods in comparing the results, keeps the test objective. Calculations for the statistical analysis is done through the use of the Statistical Package for Social Sciences (SPSS). SPSS was used due to personal experience with the software, as well as SPSS's comprehensive data mining technology and analytic applications for enhanced decision-making.

3.3.3 Component comparison

As mentioned earlier, four components were identified for evaluation on the two servers:

- Static document retrieval;
- Common Gateway Interface binaries (CGI);
- Dynamic Shared Objects (DSO) & Dynamic Linked Libraries (DLL); and
- Cached DSO & DLL.

Each of these four components will be tested at various stress levels over the three different document sizes (paragraph 3.3.1). This means that there will be 12 different performance tests, based on document type and document size.

3.3.3.1 Static document retrieval

3.3.3.1.1 *Performance comparison: Small documents*

Kolmogorov-Smirnov Z: Test for goodness of fit to the normal distribution	
H_0: Samples obtained from normal population ∴ *Use paired t test to compare samples* H_1: Samples not obtained from normal population ∴ *Use Wilcoxon matched-pairs signed rank test*	*n* = 32 Reject H_0 if $z < .23$ ($T_{32,.05}$)
Obtained values: z_{Apache}: .652 $z_{Information\ Server}$: .783	**Results:** Apache: Reject H_0 Information Server: Reject H_0
Conclusion: Neither of the two samples appear to originate from a normal distribution. Use Wilcoxon matched-pairs signed rank test	

Wilcoxon matched-pairs signed rank test

	N	Mean Rank	Sum of Ranks
Negative ranks	15[a]	14.63	219.5
Positive ranks	17[b]	18.15	308.5
Ties	0[c]		
Total	32		
Z	-.832		

a:	IIS slower than Apache
b:	IIS faster than Apache
c:	Equal performance

Two-tailed test for equality:

H_0: $\mu_{Apache} = \mu_{Information\ Server}$ H_1: $\mu_{Apache} \neq \mu_{Information\ Server}$

α: .05 Reject H_0 if $|z_T| > 1.96$

Conclusion: Accept H_0 as $|-.832| > 1.96$ fails

As we accepted H_0, it is not required to do a one-tailed test to test if one server is faster than the other.

3.3.3.1.2 *Performance comparison: Medium-sized documents*

Kolmogorov-Smirnov Z: Test for goodness of fit to the normal distribution	
H_0: Samples obtained from normal population ∴ *Use paired t test to compare samples* H_1: Samples not obtained from normal population ∴ *Use Wilcoxon matched-pairs signed rank test*	$n = 32$ Reject H_0 if $z < .23$ ($T_{32,.05}$)
Obtained values: z_{Apache}: .861 $z_{Information\ Server}$: .755	**Results:** Apache: Reject H_0 Information Server: Reject H_0
Conclusion: Neither of the two samples appear to originate from a normal distribution. Use Wilcoxon matched-pairs signed rank test	

Wilcoxon matched-pairs signed rank test

	N	Mean Rank	Sum of Ranks
Negative ranks	32[a]	16.5	528
Positive ranks	0[b]	0	0
Ties	0[c]	a:	IIS slower than Apache
Total	32	b:	IIS faster than Apache
Z	-4.937	c:	Equal performance

Two-tailed test for equality:

H_0: $\mu_{Apache} = \mu_{Information\ Server}$ H_1: $\mu_{Apache} \neq \mu_{Information\ Server}$

α: .05 Reject H_0 if $|z_T| > 1.96$

Conclusion: Reject H_0 as $|-4.937| > 1.96$ holds

One-tailed test - Apache provides a higher output than Information Server

H_0: $\mu_{Apache} = \mu_{Information\ Server}$ H_1: $\mu_{Apache} > \mu_{Information\ Server}$

α: .05 Reject H_0 if $|z_T| > 1.6449$

Conclusion: Reject H_0 as $|-4.937| > 1.6449$ holds

For medium-sized static documents, it was found that Apache was the better server under load.

3.3.3.1.3 *Performance comparison: Large documents*

Kolmogorov-Smirnov Z: Test for goodness of fit to the normal distribution	
H_0: Samples obtained from normal population ∴ *Use paired t test to compare samples* H_1: Samples not obtained from normal population ∴ *Use Wilcoxon matched-pairs signed rank test*	$n = 32$ Reject H_0 if $z < .23$ ($T_{32,.05}$)
Obtained values: z_{Apache}: .665 $z_{Information\ Server}$: .913	**Results:** Apache: Reject H_0 Information Server: Reject H_0
Conclusion: Neither of the two samples appear to originate from a normal distribution. Use Wilcoxon matched-pairs signed rank test	

Wilcoxon matched-pairs signed rank test

	N	Mean Rank	Sum of Ranks
Negative ranks	31[a]	17	527
Positive ranks	1[b]	1	1
Ties	0[c]	a:	IIS slower than Apache
Total	32	b:	IIS faster than Apache
Z	-4.918	c:	Equal performance

Two-tailed test for equality:

H_0: $\mu_{Apache} = \mu_{Information\ Server}$ H_1: $\mu_{Apache} \neq \mu_{Information\ Server}$

α: .05 Reject H_0 if $|z_T| > 1.96$

Conclusion: Reject H_0 as $|-4.918| > 1.96$ holds

One-tailed test - Apache provides a higher output than Information Server

H_0: $\mu_{Apache} = \mu_{Information\ Server}$ H_1: $\mu_{Apache} > \mu_{Information\ Server}$

α: .05 Reject H_0 if $|z_T| > 1.6449$

Conclusion: Reject H_0 as $|-4.918| > 1.6449$ holds

As with the medium-sized documents, Apache performs better than Information Server.

3.3.3.2 Common Gateway Interface binaries

3.3.3.2.1 *Performance comparison: Small documents*

Kolmogorov-Smirnov Z: Test for goodness of fit to the normal distribution	
H_0: Samples obtained from normal population ∴ *Use paired t test to compare samples* H_1: Samples not obtained from normal population ∴ *Use Wilcoxon matched-pairs signed rank test*	$n = 32$ Reject H_0 if $z < .23$ ($T_{32,.05}$)
Obtained values: z_{Apache}: 1.179 $z_{Information\ Server}$: .632	**Results:** Apache: Reject H_0 Information Server: Reject H_0
Conclusion: Neither of the two samples appear to originate from a normal distribution. Use Wilcoxon matched-pairs signed rank test	

Wilcoxon matched-pairs signed rank test

	N	Mean Rank	Sum of Ranks
Negative ranks	32[a]	16.5	528
Positive ranks	0[b]	0	0
Ties	0[c]	a:	IIS slower than Apache
Total	32	b:	IIS faster than Apache
Z	-4.937	c:	Equal performance

Two-tailed test for equality:

H_0: $\mu_{Apache} = \mu_{Information\ Server}$ H_1: $\mu_{Apache} \neq \mu_{Information\ Server}$

α: .05 Reject H_0 if $|z_T| > 1.96$

Conclusion: Reject H_0 as $|-4.937| > 1.96$ holds

One-tailed test - Apache provides a higher output than Information Server

H_0: $\mu_{Apache} = \mu_{Information\ Server}$ H_1: $\mu_{Apache} > \mu_{Information\ Server}$

α: .05 Reject H_0 if $|z_T| > 1.6449$

Conclusion: Reject H_0 as $|-4.937| > 1.6449$ holds

Apache performs better serving small documents from a CGI binary than Information Server.

3.3.3.2.2 *Performance comparison: Medium-sized documents*

Kolmogorov-Smirnov Z: Test for goodness of fit to the normal distribution

H_0: Samples obtained from normal population ∴ *Use paired t test to compare samples* H_1: Samples not obtained from normal population ∴ *Use Wilcoxon matched-pairs signed rank test*	$n = 32$ Reject H_0 if $z < .23$ ($T_{32,.05}$)
Obtained values: z_{Apache}: 1.38 $z_{Information\ Server}$: .519	**Results:** Apache: Reject H_0 Information Server: Reject H_0
Conclusion: Neither of the two samples appear to originate from a normal distribution. Use Wilcoxon matched-pairs signed rank test	

Wilcoxon matched-pairs signed rank test

	N	Mean Rank	Sum of Ranks
Negative ranks	32[a]	16.5	528
Positive ranks	0[b]	0	0
Ties	0[c]	a:	IIS slower than Apache
Total	32	b:	IIS faster than Apache
Z	-4.937	c:	Equal performance

Two-tailed test for equality:

H_0: $\mu_{Apache} = \mu_{Information\ Server}$ H_1: $\mu_{Apache} \neq \mu_{Information\ Server}$

α: .05 Reject H_0 if $|z_T| > 1.96$

Conclusion: Reject H_0 as $|-4.937| > 1.96$ holds

One-tailed test - Apache provides a higher output than Information Server

H_0: $\mu_{Apache} = \mu_{Information\ Server}$ H_1: $\mu_{Apache} > \mu_{Information\ Server}$

α: .05 Reject H_0 if $|z_T| > 1.6449$

Conclusion: Reject H_0 as $|-4.937| > 1.6449$ holds

Apache shows to outperform Information Server when serving medium-sized documents through a CGI binary.

3.3.3.2.3 *Performance comparison: Large documents*

Kolmogorov-Smirnov Z: Test for goodness of fit to the normal distribution	
H_0: Samples obtained from normal population ∴ *Use paired t test to compare samples* H_1: Samples not obtained from normal population ∴ *Use Wilcoxon matched-pairs signed rank test*	*n* = 32 Reject H_0 if $z < .23$ ($T_{32,.05}$)
Obtained values: z_{Apache}: 1.311 $z_{Information\ Server}$: 1.046	**Results:** Apache: Reject H_0 Information Server: Reject H_0
Conclusion: Neither of the two samples appear to originate from a normal distribution. Use Wilcoxon matched-pairs signed rank test	

Wilcoxon matched-pairs signed rank test

	N	Mean Rank	Sum of Ranks
Negative ranks	12[a]	13.96	167.5
Positive ranks	20[b]	18.02	360.5
Ties	0[c]	a:	IIS slower than Apache
Total	32	b:	IIS faster than Apache
Z	-1.805	c:	Equal performance

Two-tailed test for equality:

H_0: $\mu_{Apache} = \mu_{Information\ Server}$ H_1: $\mu_{Apache} \neq \mu_{Information\ Server}$

α: .05 Reject H_0 if $|z_T| > 1.96$

Conclusion: Accept H_0 as $|-1.805| > 1.96$ fails

Again, no two-tailed test is required, as the test states that there is no significant performance difference between the two servers.

3.3.3.3 Dynamic Shared Objects and Dynamic Linked Libraries

3.3.3.3.1 *Performance comparison: Small documents*

Kolmogorov-Smirnov Z: Test for goodness of fit to the normal distribution	
H_0: Samples obtained from normal population ∴ *Use paired t test to compare samples* H_1: Samples not obtained from normal population ∴ *Use Wilcoxon matched-pairs signed rank test*	*n* = 32 Reject H_0 if z < .23 ($T_{32,.05}$)
Obtained values: z_{Apache}: 2.65 $z_{Information\ Server}$: 1.85	**Results:** Apache: Reject H_0 Information Server: Reject H_0
Conclusion: Neither of the two samples appear to originate from a normal distribution. Use Wilcoxon matched-pairs signed rank test	

Wilcoxon matched-pairs signed rank test

	N	Mean Rank	Sum of Ranks
Negative ranks	16[a]	19.38	310
Positive ranks	16[b]	13.63	218
Ties	0[c]	a:	IIS slower than Apache
Total	32	b:	IIS faster than Apache
Z	-.86	c:	Equal performance

Two-tailed test for equality:

H_0: $\mu_{Apache} = \mu_{Information\ Server}$ H_1: $\mu_{Apache} \neq \mu_{Information\ Server}$

α: .05 Reject H_0 if $|z_T| > 1.96$

Conclusion: Accept H_0 as $|-.86| > 1.96$ fails

The outcome of this test finds again no significant differences in the output levels of Apache and Information Server.

3.3.3.3.2 *Performance comparison: Medium-sized documents*

Kolmogorov-Smirnov Z: Test for goodness of fit to the normal distribution	
H_0: Samples obtained from normal population ∴ *Use paired t test to compare samples* H_1: Samples not obtained from normal population ∴ *Use Wilcoxon matched-pairs signed rank test*	$n = 32$ Reject H_0 if $z < .23$ $(T_{32,.05})$
Obtained values: z_{Apache}: .825 $z_{Information\ Server}$: .701	**Results:** Apache: Reject H_0 Information Server: Reject H_0
Conclusion: Neither of the two samples appear to originate from a normal distribution. Use Wilcoxon matched-pairs signed rank test	

Wilcoxon matched-pairs signed rank test

	N	Mean Rank	Sum of Ranks
Negative ranks	25[a]	18.12	453
Positive ranks	7[b]	10.71	75
Ties	0[c]	a:	IIS slower than Apache
Total	32	b:	IIS faster than Apache
Z	-3.534	c:	Equal performance

Two-tailed test for equality:

H_0: $\mu_{Apache} = \mu_{Information\ Server}$ H_1: $\mu_{Apache} \neq \mu_{Information\ Server}$

α: .05 Reject H_0 if $|z_T| > 1.96$

Conclusion: Reject H_0 as $|-3.534| > 1.96$ holds

One-tailed test - Apache provides a higher output than Information Server

H_0: $\mu_{Apache} = \mu_{Information\ Server}$ H_1: $\mu_{Apache} > \mu_{Information\ Server}$

α: .05 Reject H_0 if $|z_T| > 1.6449$

Conclusion: Reject H_0 as $|-3.534| > 1.6449$ holds

From the test results, it was found that Apache can serve more requests than Information Server when medium-sized documents are loaded through DSO/DLL's.

3.3.3.3.3 *Performance comparison: Large documents*

Kolmogorov-Smirnov Z: Test for goodness of fit to the normal distribution	
H_0: Samples obtained from normal population ∴ *Use paired t test to compare samples* H_1: Samples not obtained from normal population ∴ *Use Wilcoxon matched-pairs signed rank test*	$n = 32$ Reject H_0 if $z < .23$ $(T_{32,.05})$
Obtained values: z_{Apache}: .475 $z_{Information\ Server}$: .649	**Results:** Apache: Reject H_0 Information Server: Reject H_0
Conclusion: Neither of the two samples appear to originate from a normal distribution. Use Wilcoxon matched-pairs signed rank test	

Wilcoxon matched-pairs signed rank test

	N	Mean Rank	Sum of Ranks
Negative ranks	5[a]	10.7	53.5
Positive ranks	27[b]	17.57	474.5
Ties	0[c]	a:	IIS slower than Apache
Total	32	b:	IIS faster than Apache
Z	-3.937	c:	Equal performance

Two-tailed test for equality:

H_0: $\mu_{Apache} = \mu_{Information\ Server}$ H_1: $\mu_{Apache} \neq \mu_{Information\ Server}$

α: .05 Reject H_0 if $|z_T| > 1.96$

Conclusion: Reject H_0 as $|-3.937| > 1.96$ holds

One-tailed test - Information Server provides a higher output than Apache

H_0: $\mu_{Apache} = \mu_{Information\ Server}$ H_1: $\mu_{Apache} < \mu_{Information\ Server}$

α: .05 Reject H_0 if $|z_T| > 1.6449$

Conclusion: Reject H_0 as $|-3.937| > 1.6449$ holds

When requesting large documents through a DSO/DLL, Information Server is capable of serving more requests than Apache at the same increasing load.

3.3.3.4 Cached documents from Dynamic Shared Objects and Dynamic Linked Libraries

3.3.3.4.1 *Performance comparison: Small documents*

Kolmogorov-Smirnov Z: Test for goodness of fit to the normal distribution	
H_0: Samples obtained from normal population ∴ *Use paired t test to compare samples* H_1: Samples not obtained from normal population ∴ *Use Wilcoxon matched-pairs signed rank test*	$n = 32$ Reject H_0 if $z < .23$ ($T_{32,.05}$)
Obtained values: z_{Apache}: 1.289 $z_{Information\ Server}$: .789	**Results:** Apache: Reject H_0 Information Server: Reject H_0
Conclusion: Neither of the two samples appear to originate from a normal distribution. Use Wilcoxon matched-pairs signed rank test	

Wilcoxon matched-pairs signed rank test

	N	Mean Rank	Sum of Ranks
Negative ranks	26[a]	13.77	358
Positive ranks	6[b]	28.33	170
Ties	0[c]	a:	IIS slower than Apache
Total	32	b:	IIS faster than Apache
Z	-1.758	c:	Equal performance

Two-tailed test for equality:

H_0: $\mu_{Apache} = \mu_{Information\ Server}$ H_1: $\mu_{Apache} \neq \mu_{Information\ Server}$

α: .05 Reject H_0 if $|z_T| > 1.96$

Conclusion: Accept H_0 as $|-1.758| > 1.96$ fails

No significant difference was found between the two servers, yielding an equal observed output at a .95 confidence level.

3.3.3.4.2 *Performance comparison: Medium-sized documents*

Kolmogorov-Smirnov Z: Test for goodness of fit to the normal distribution	
H_0: Samples obtained from normal population ∴ *Use paired t test to compare samples* H_1: Samples not obtained from normal population ∴ *Use Wilcoxon matched-pairs signed rank test*	$n = 32$ Reject H_0 if $z < .23$ ($T_{32,.05}$)
Obtained values: z_{Apache}: 1.188 $z_{Information\ Server}$: .605	**Results:** Apache: Reject H_0 Information Server: Reject H_0
Conclusion: Neither of the two samples appear to originate from a normal distribution. Use Wilcoxon matched-pairs signed rank test	

Wilcoxon matched-pairs signed rank test

	N	Mean Rank	Sum of Ranks
Negative ranks	31[a]	17	527
Positive ranks	1[b]	1	1
Ties	0[c]	a:	IIS slower than Apache
Total	32	b:	IIS faster than Apache
Z	-4.918	c:	Equal performance

Two-tailed test for equality:

H_0: $\mu_{Apache} = \mu_{Information\ Server}$ H_1: $\mu_{Apache} \neq \mu_{Information\ Server}$

α: .05 Reject H_0 if $|z_T| > 1.96$

Conclusion: Reject H_0 as $|-4.918| > 1.96$ holds

One-tailed test - Apache provides a higher output than Information Server

H_0: $\mu_{Apache} = \mu_{Information\ Server}$ H_1: $\mu_{Apache} > \mu_{Information\ Server}$

α: .05 Reject H_0 if $|z_T| > 1.6449$

Conclusion: Reject H_0 as $|-4.918| > 1.6449$ holds

When serving medium-sized documents from the DSO/DLL's memory, Apache is significantly faster than Information Server.

3.3.3.4.3 *Performance comparison: Large documents*

Kolmogorov-Smirnov Z: Test for goodness of fit to the normal distribution	
H_0: Samples obtained from normal population ∴ *Use paired t test to compare samples* H_1: Samples not obtained from normal population ∴ *Use Wilcoxon matched-pairs signed rank test*	*n* = 32 Reject H_0 if $z < .23$ ($T_{32,.05}$)
Obtained values: z_{Apache}: .479 $z_{Information\ Server}$: 2.194	**Results:** Apache: Reject H_0 Information Server: Reject H_0
Conclusion: Neither of the two samples appear to originate from a normal distribution. Use Wilcoxon matched-pairs signed rank test	

Wilcoxon matched-pairs signed rank test

	N	Mean Rank	Sum of Ranks
Negative ranks	5[a]	3	15
Positive ranks	27[b]	19	513
Ties	0[c]	a:	IIS slower than Apache
Total	32	b:	IIS faster than Apache
Z	-4.656	c:	Equal performance

Two-tailed test for equality:

H_0: $\mu_{Apache} = \mu_{Information\ Server}$ H_1: $\mu_{Apache} \neq \mu_{Information\ Server}$

α: .05 Reject H_0 if $|z_T| > 1.96$

Conclusion: Reject H_0 as $|-4.656| > 1.96$ holds

One-tailed test - Information Server provides a higher output than Apache

H_0: $\mu_{Apache} = \mu_{Information\ Server}$ H_1: $\mu_{Apache} < \mu_{Information\ Server}$

α: .05 Reject H_0 if $|z_T| > 1.6449$

Conclusion: Reject H_0 as $|-4.656| > 1.6449$ holds

Information Server seems to regain the upper hand on Apache in serving large documents from its DSO/DLL memory cache.

3.3.3.5 Performance comparison summary

Based on the preceding tests, the following grid serves as a summary, showing the faster of the two servers in the respective areas:

	Small	Medium	Large
Static	-	Apache	Apache
CGI	Apache	Apache	-
DSO	-	Apache	Information Server
Cached DSO	-	Apache	Information Server

Table 3.1 Web server performance comparison summary

A clear deduction that can be made from this summary is that Apache appears to be faster than Information Server in more cases than where it is slower. By using this profile, certain assumptions can be made of both servers' methodologies in architecture and their real-world applications.

3.3.4 Performance results

3.3.4.1 Static document retrieval

3.3.4.1.1 *Small static documents*

Figure 3.2 shows the different performance profiles of the two servers in the event of an increasing load. Statistically, it was found that neither of the two servers were significantly faster than the other. If one looks at the performance charts, an interesting trend is observed. Even though the two servers perform at relatively the same output levels, Apache's output deteriorates much quicker than that of Information Server. When the linear trend lines are compared, it is found that the two lines meet around a load level of 360 clients, after which, Information Server outperforms Apache. This shows that Information Server's internal caching, both at web server and operating system level, is much better than Apache's at a higher load level for small files. Information Server does not remain at a high output level forever though. Information Server's output declines as well, but at a slower rate.

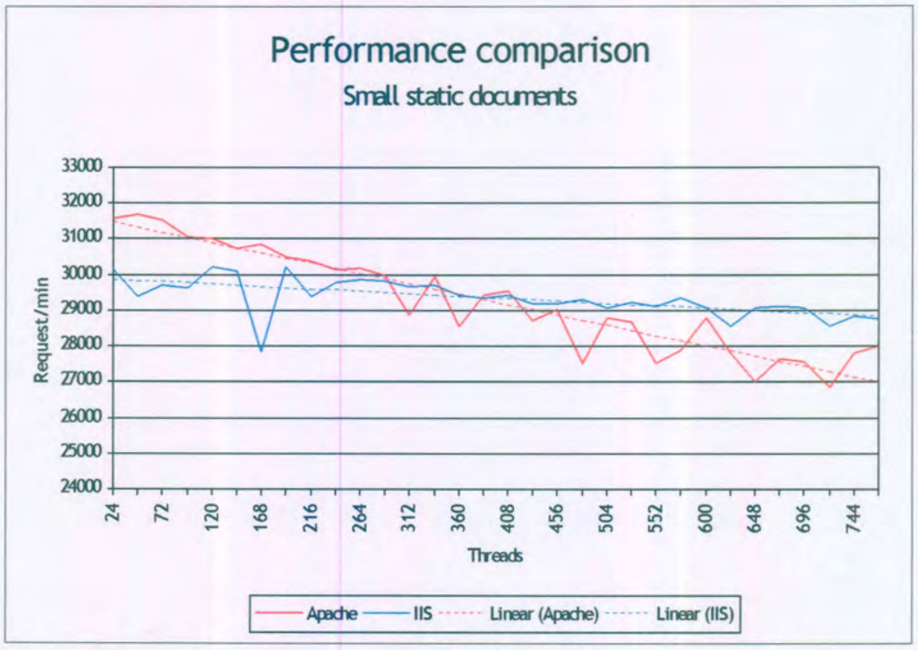


Figure 3.2 Performance comparison: Small static documents

Both servers seem to be able to serve high numbers of clients with small files (around 15-20K). It is however recommended that Information Server be used where the number of simultaneous connections exceed 768. Where the number of simultaneous connections is under 360, Apache should make the better server.

3.3.4.1.2 *Medium-sized static documents*

From Figure 3.3, the difference in output performance can be clearly seen. Apache is significantly faster than Information Server. In contrast to the previous chart, Information Server shows a greater decrease in performance levels. This suggests that Apache's caching mechanisms are better configured towards medium-sized files (100-200K) than that of Information Server. Even though Apache's linear trend suggests an ever increasing output capability, one could expect a decrease in its output level at a much higher stress level.

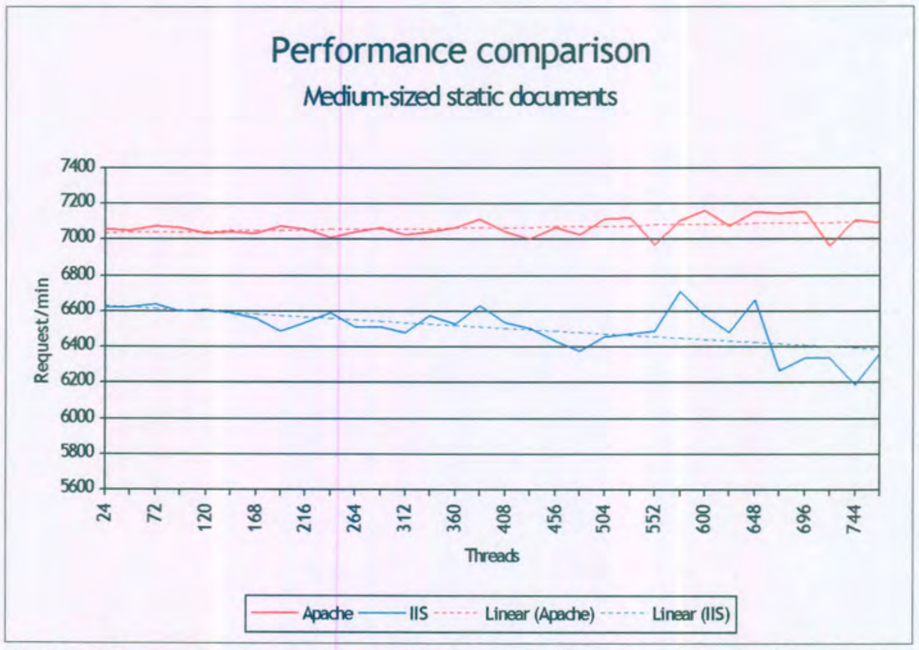


Figure 3.3 Performance comparison: Medium-sized static documents

3.3.4.1.3 *Large static documents*

Neither of the two servers was able to keep their output at a constant level. Again, Information Server showed lower performance, and a faster decline in output. Both servers showed drastic declines in output, and at the last point in the chart, Information Server could only serve around 60% of the requests that Apache could.

Apache would therefore be much more effective when applied as an online document archive, in particular where the document sizes exceed 500K.

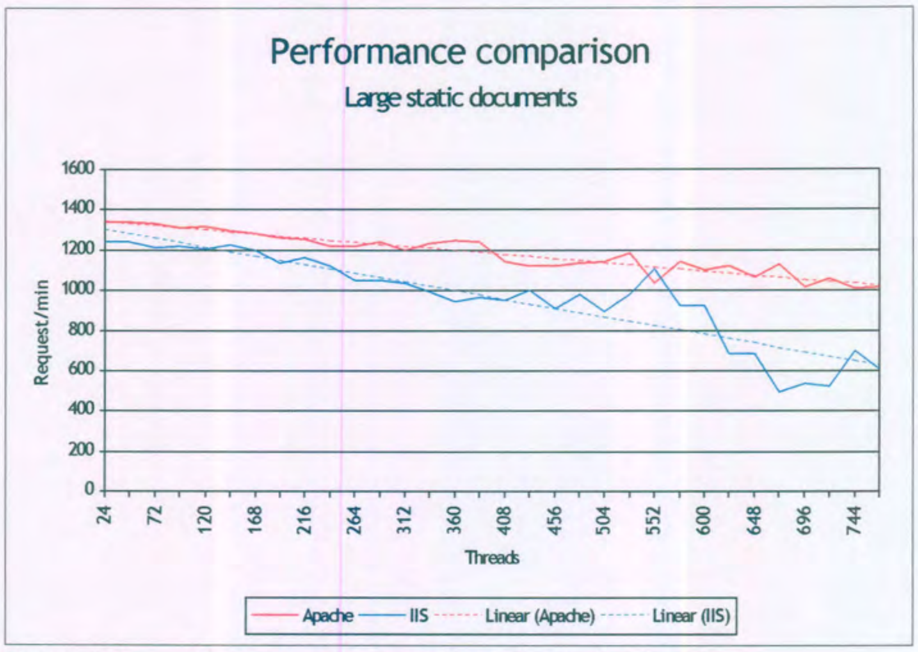


Figure 3.4 Performance comparison: Large documents

3.3.4.1.4 *Summary*

The Internet as an academic network was founded on document archives and libraries. Today, with the arrival of Internet Commerce things have changed slightly, but the old values and applications still remain. Many companies have resorted to intranet servers to function as document flow management systems. Sun's Group Manager: Portal Solutions, Steve Oglethorpe says the following on Enterprise Portals (2002:www.sun.com)

> "Across horizontal applications lie numerous portal-based services, including creation and maintenance of a data repository, search-and-retrieval capability, and activity tracking."

He continues highlighting the return on investment (ROI) of these kinds of technologies:

> "Portals enabling delivery of services on demand will become more necessary, and in the current economic environment, ROI justification will be required every step of the

way. Increasing the value of content and services through improved usability and reduced cost per transaction breeds increased business quickness and efficiency. This relationship is at the core of the dramatic ROI of enterprise information portal deployments. A successful portal brings rationality to an enterprise's data and IT resources, communications and operations efficiency, and flexibility for whatever the future may bring."

By applying Apache as a data repository for Intranets, Extranets and Enterprise Portals, the ROI could be much higher than when Information Server is used, purely based on its ability to serve static documents at a much higher rate than his competitor. It is equally important to consider the low cost of the Linux/Apache architecture.

3.3.4.2 Using Common Gateway Interface(CGI) binaries to serve Internet documents

The test involves a CGI binary that resides on the web server and loads a file from the underlying filesystem which is then served to the client. By using CGI binaries on a web server, one is capable of interacting with databases and transaction systems without having the client to directly interact with these systems. This is a popular technology for performing e-commerce transactions.

3.3.4.2.1 *Small documents*

The performance charts reiterate the statistic findings that Apache is faster than Information Server in using CGI-binaries to serve small documents from a web server. As CGI's have to be loaded from the filesystem, it can place a big load on the operating system. Furthermore, if the filesystem is too slow to load the CGI, it will slow down the web server. What is seen here is that Apache is much better prepared to serve requests through CGI binaries. Both servers show a slight slowdown at higher loads, but neither of these are critical.

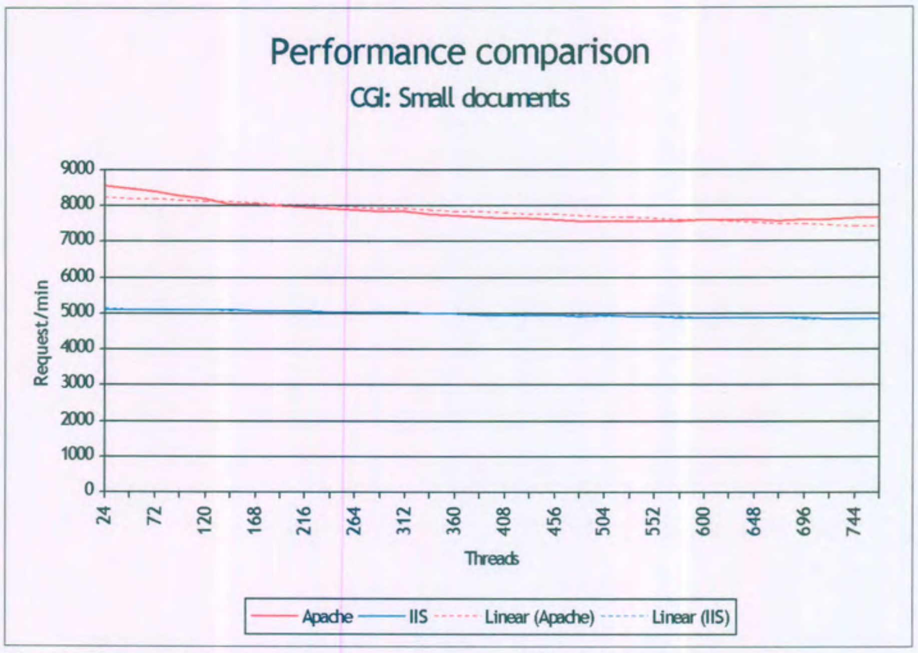


Figure 3.5 Performance comparison: CGI: Small documents

3.3.4.2.2 *Medium-sized documents*

A similar profile is observed with medium-sized documents, with the exception of the actual output figures. When the performance of CGI's with medium-sized documents are compared with small documents, the average observed difference between the two servers fell from 2,500 to only about 500 requests served per minute. Both servers show decreased performance at higher levels, with Apache's performance decreasing faster than Information Server.

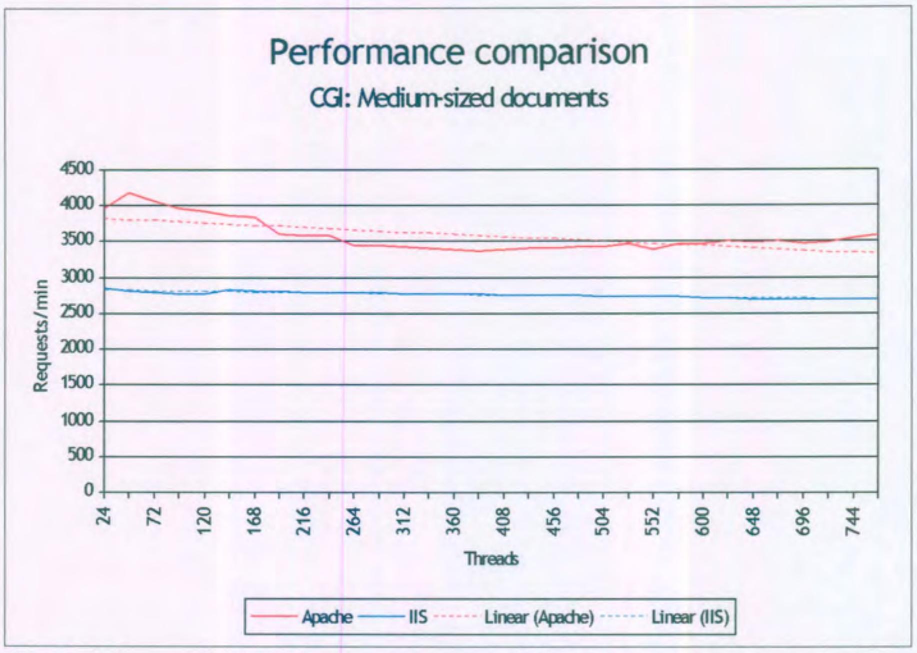


Figure 3.6 Performance comparison: CGI: Medium-sized documents

3.3.4.2.3 *Large documents*

The statistical analysis on the two performance series found no significant difference between them (Figure 3.7). Apache experienced a bigger decline in service levels than Information Server at high load. At a connection number of around 300 clients, Apache's linear trend crosses that of Information Server and goes down drastically. This profile contradicts the previous two seen (small and medium documents) where both servers had a relatively slow decline in output. At the start of the test, Apache could serve almost 200 more requests than Information Server. At the highest load recorded (768 clients), Apache could only manage half of what Information Server was capable of serving.

When the load on the server is low, Apache should perform better than Information Server. However, when the server load exceeds 300 concurrent connections, Information Server should be used due to its ability to handle high traffic better than Apache for this application.

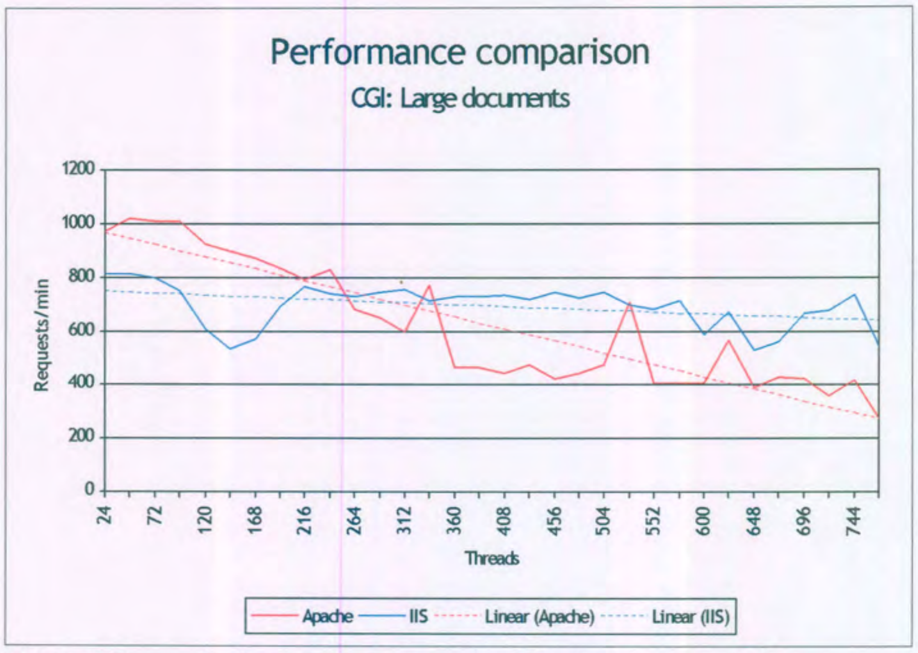


Figure 3.7 Performance comparison: CGI: Large documents

3.3.4.2.4 *Summary*

CGI is an attractive option to use for dynamic content creation on web sites, as it is relatively fast and simple to deploy. In comparison with ISAPI DLL's and Apache DSO's, it is much easier to deploy and configure. These modules can also be replaced without restarting the server, as is the case with DLL's or DSO's. It is expected that CGI's cannot produce the same output as the server extensions (DLL & DSO). This is due to the fact that CGI's are not cached extensions of the server, but use the server as execution platform and terminates after its completion of the request. Every request is serviced by a separate application instance (Borland, 27-7, 2001).

Where CGI's are required for server development, Apache is recommended only if the response size is relatively small. For more complicated processing and large output files (>500K) at high load, Information Server is recommended. This is based on the higher output levels obtained from the CGI tests on both servers.

3.3.4.3 Dynamic Shared Objects (Apache DSO) and Dynamic Linked Libraries (ISAPI DLL)

Using DSO's and DLL's rather than CGI's are preferred where high server load is expected. Because DSO's and DLL's are cached as extensions to the web server, it is able to respond to incoming requests much faster, using less of the operating system's resources. This test involved the libraries to load the exact same files from disk as in the previous two test sections at the same increasing load. The files will not be cached within the server extension.

3.3.4.3.1 *Small documents*

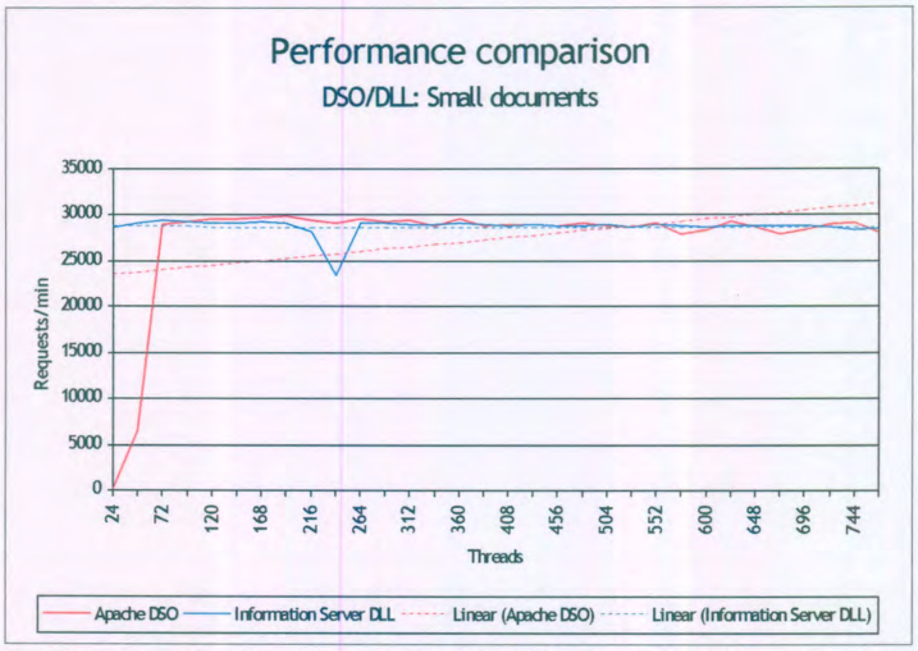


Figure 3.8 Performance comparison: DSO/DLL: Small documents

From Figure 3.8, it is easy to see why the two servers were found to perform at the same output levels. With the exception of Apache's slow start which can be attributed to thread initialisation at the test's startup, Apache and Information Server performs at approximately the same level throughout the test. Neither of these two servers show any significant slowdown to the end of the test and seems to be able to remain at their respective output levels in serving the files.

3.3.4.3.2 *Medium-sized documents*

The two servers have a similar performance pattern for serving medium-sized documents through DSO/DLL libraries. The two servers do not perform at the same output level, with Apache being significantly faster. Information Server shows a more definite decrease in performance than that of Apache, but this is only marginally in terms of the test data.

Apache had a slower start again but recovered to such a level to outperform Information Server through the rest of the test.

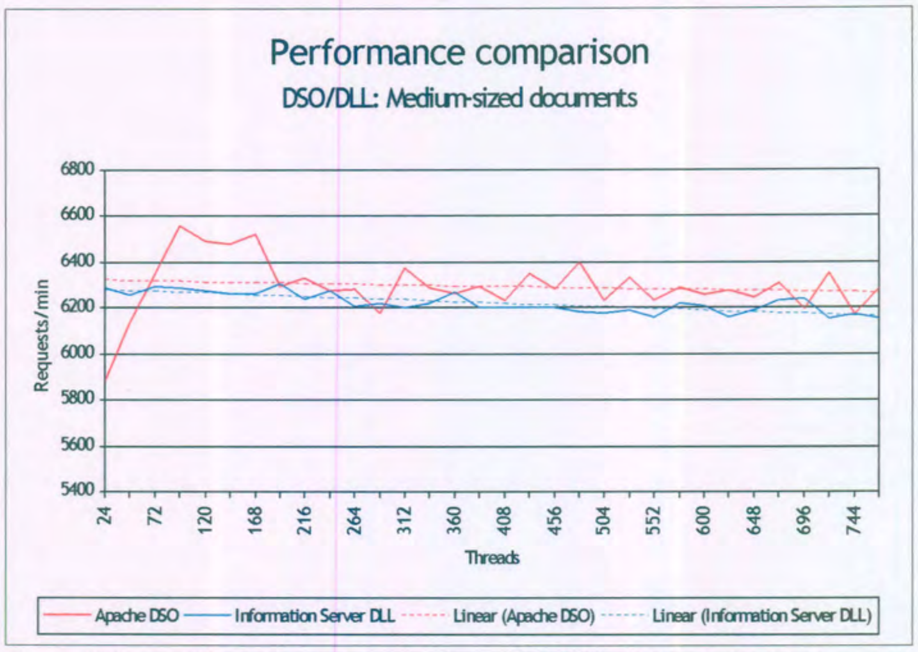


Figure 3.9 Performance comparison: DSO/DLL:Medium-sized documents

3.3.4.3.3 *Large documents*

Information Server was found to serve large documents faster than Apache (Figure 3.10). The servers show the same output profile, though Apache's is at a lower request level. Both servers kept their output levels at a constant level, without any serious deterioration of service. A constant output by both servers suggests good stability on both sides, emphasising that both operating systems are tuned to perform strenuous activity at high loads and for long periods.

3.3.4.3.4 *Summary*

DSO/DLL libraries are used to extend web servers. In contrast to CGI binaries, DSO/DLL libraries do not use the web server as execution platform, but extends its functionality by attaching itself to the web server's core. In essence, it becomes part of the server. By being loaded into the web server at startup, it has the advantage of being ready and waiting as soon as the first request hits the server. CGI's have to be loaded for every request posted to it, slowing down the server. The test was performed to measure the dynamic module's ability to load files from the filesystem, as it was already memory-resident itself.

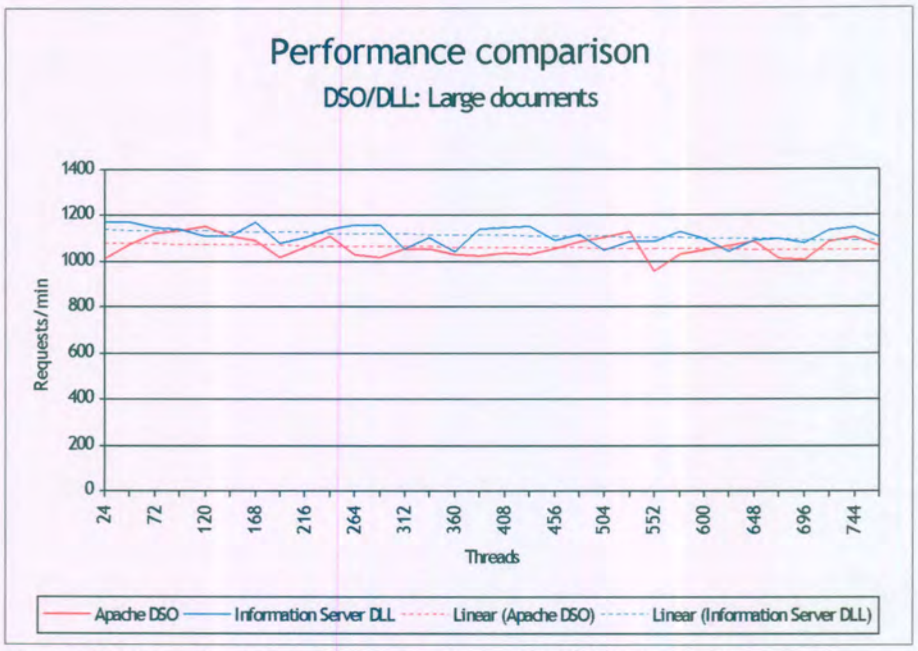


Figure 3.10 Performance comparison: DSO/DLL: Large documents

3.3.4.4 Cached documents from Dynamic Shared Objects (Apache DSO) and Dynamic Linked Libraries (ISAPI DLL)

Compared to the previous test, this test focusses on the DSO/DLL libraries' ability to serve documents to clients that are already cached in memory, as opposed to loading it first from disk and then serving it. As access to memory is faster than access to disk (Stallings,1995:22), the available documents are cached in memory and served from there. This has the advantage of bypassing the slower physical storage system, thereby speeding up document serving to clients.

3.3.4.4.1 *Small documents*

Neither of the two servers was found to be significantly faster than the other. Figure 3.11 confirms this, as the two performance lines are very close to each other. When compared with the non-cached tests, the cached-model performs better. This reiterates the argument for caching server documents inside dynamic modules, rather than loading them from file.

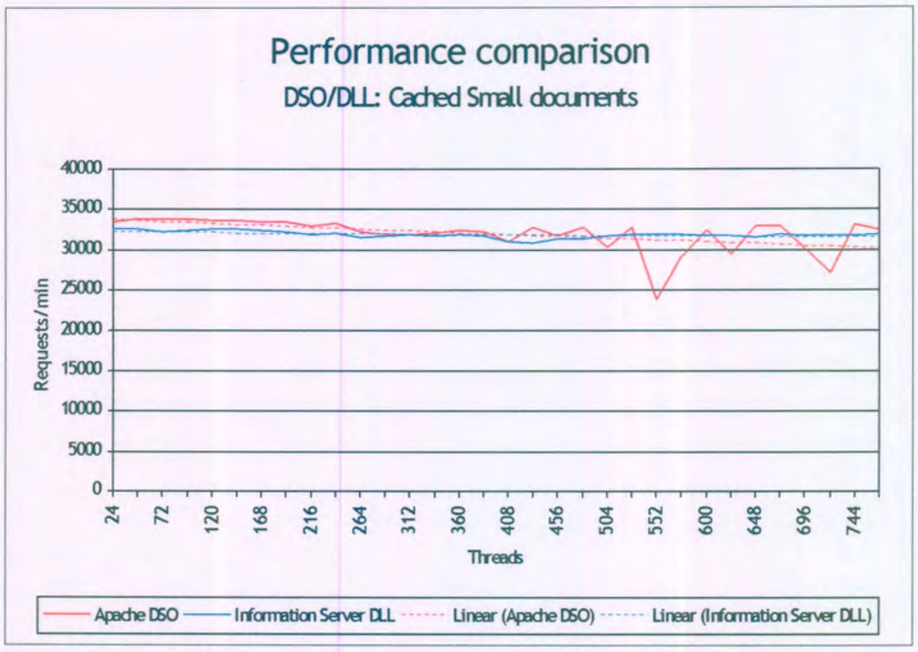


Figure 3.11 Performance comparison: DSO/DLL: Cached small documents

3.3.4.4.2 *Medium-sized files*

Figure 3.12 shows that both servers have taken a huge performance hit due to the bigger document size, if compared to Figure 3.11. It was found that Apache is faster than Information Server in this test, which is also shown in the chart. Between the servers there is around a 500-request per minute difference (7%). What is interesting to note at this point is that Apache has performed the best in all tests on this document size. This suggests that Apache is better suited to serve documents that are in this size range (around 100K).

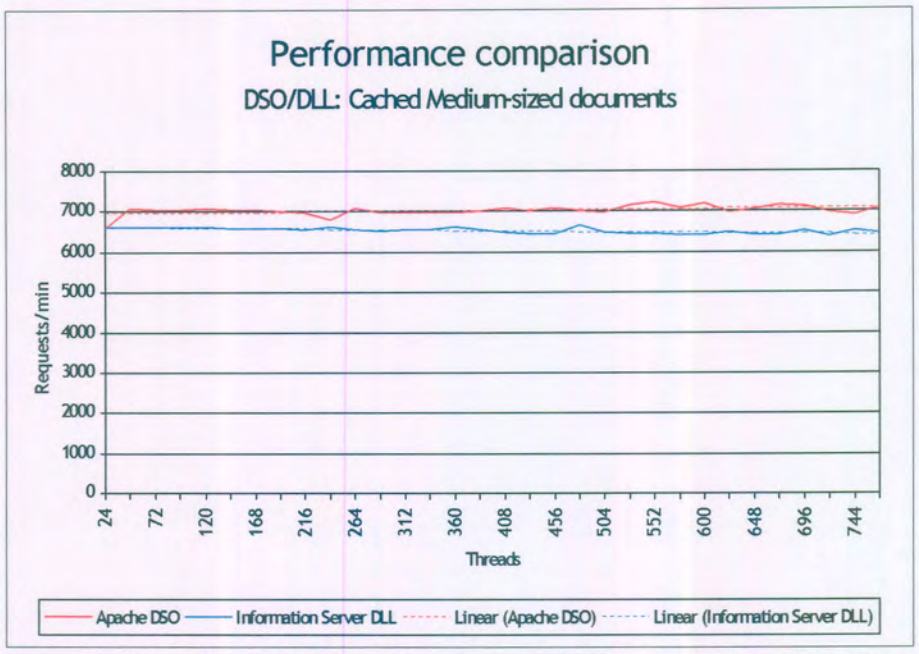


Figure 3.12 Performance comparison: DSO/DLL Cached medium-sized documents

3.3.4.4.3 *Large documents*

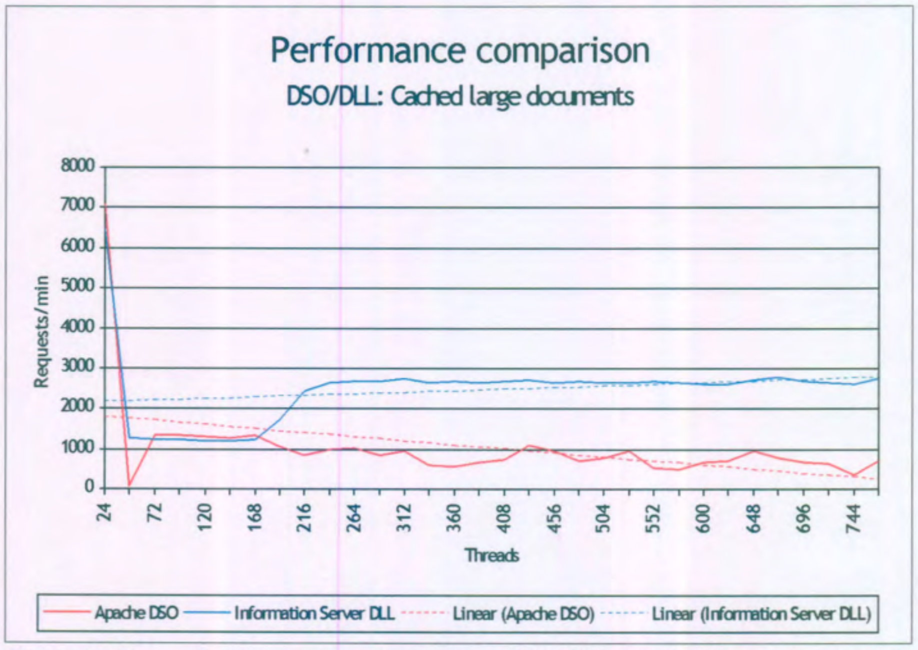


Figure 3.13 Performance comparison: DSO/DLL: Cached large documents

The last test found Information Server to be the faster of the two. Apache is not suitable for serving large documents from memory through a Dynamic Shared Object. As the linear trend lines suggest, Apache's performance is decreasing rapidly to the end of the test, which makes it unsuitable for serving documents in this fashion at a high load. Information Server on the other hand, is more than capable of handling the high number of requests, handling more than three times as much as Apache at the stress level of 768 clients. By loading highly requested documents into the Information Server's ISAPI DLL memory, one would be able to serve clients much better than when Apache and Dynamic Shared Objects are used in the same fashion.

3.3.4.4.4 *Summary*

Caching is present in almost all aspects of Computer Science and software development based on its proven success. For Internet Development, this concept is even more true. By serving cached documents from a server extension's memory enables the server to perform better while using fewer system resources. Dynamic Shared Objects and ISAPI DLL's enable this as they can be loaded into the web server and remain memory-resident as long as the server is running. This has countless applications in the web development field. Cached modules are effectively used as high performance web server modules on several web sites. The Borland Delphi 6 Developer's Guide says the following on the deployment of DLL's:

> "Creating a Web server application as a DLL reduces system load and resource use by reducing the number of processes and disk accesses necessary to service an individual request" (Borland, 2001:27-6)

3.3.5 **Platform-specific comparison**

In order to get a good idea of the different server modules' performance, the tests in this chapter is supplemented by vertical comparisons as well. With these tests, the three different dynamic modules is compared to find the best performing model. The previous sections have already showed that none of the samples originate from a normal distribution, hence all performance tests will be performed through nonparametric tests, using the Wilcoxon matched-pairs signed rank test.

3.3.5.1 Apache

3.3.5.1.1 *Small documents*

Wilcoxon matched-pairs signed rank test - CGI vs DSO

	N	Mean Rank	Sum of Ranks
Negative ranks	2[a]	1.5	3
Positive ranks	30[b]	17.5	525
Ties	0[c]		
Total	32		
Z	-4.88		

a: DSO slower than CGI

b: DSO faster than CGI

c: Equal performance

One-tailed test - DSO faster than CGI

H_0: $\mu_{CGI} = \mu_{DSO}$ H_1: $\mu_{CGI} < \mu_{DSO}$

α: .05 Reject H_0 if $|z_T| > 1.6449$

Conclusion: Reject H_0 as $|-4.88| > 1.6449$ holds

Wilcoxon matched-pairs signed rank test - DSO vs Cached DSO

	N	Mean Rank	Sum of Ranks
Negative ranks	2[a]	17.5	35
Positive ranks	30[b]	16.43	493
Ties	0[c]		
Total	32		
Z	-4.282		

a: Cached DSO slower than DSO

b: Cached DSO faster than DSO

c: Equal performance

One-tailed test - Cached DSO faster than DSO

H_0: $\mu_{DSO} = \mu_{Cached\ DSO}$ H_1: $\mu_{DSO} < \mu_{Cached\ DSO}$

α: .05 Reject H_0 if $|z_T| > 1.6449$

Conclusion: Reject H_0 as $|-4.282| > 1.6449$ holds

It was found that cached DSO's deliver the best output for small documents.

3.3.5.1.2 *Medium-sized documents*

Wilcoxon matched-pairs signed rank test - CGI vs DSO

	N	Mean Rank	Sum of Ranks
Negative ranks	0[a]	0	0
Positive ranks	32[b]	16.5	528
Ties	0[c]		
Total	32		
Z	-4.937		

a: DSO slower than CGI

b: DSO faster than CGI

c: Equal performance

One-tailed test - DSO faster than CGI

H_0: $\mu_{CGI} = \mu_{DSO}$ H_1: $\mu_{CGI} < \mu_{DSO}$

α: .05 Reject H_0 if $|z_T| > 1.6449$

Conclusion: Reject H_0 as $|-4.937| > 1.6449$ holds

Wilcoxon matched-pairs signed rank test - DSO vs Cached DSO

	N	Mean Rank	Sum of Ranks
Negative ranks	0[a]	0	0
Positive ranks	32[b]	16.5	528
Ties	0[c]		
Total	32		
Z	-4.937		

a: Cached DSO slower than DSO

b: Cached DSO faster than DSO

c: Equal performance

One-tailed test - Cached DSO faster than DSO

H_0: $\mu_{DSO} = \mu_{Cached\ DSO}$ H_1: $\mu_{DSO} < \mu_{Cached\ DSO}$

α: .05 Reject H_0 if $|z_T| > 1.6449$

Conclusion: Reject H_0 as $|-4.937| > 1.6449$ holds

Again, as with small documents, it was found that cached DSO delivers better performance than the other two technologies.

3.3.5.1.3 *Large documents*

Wilcoxon matched-pairs signed rank test - CGI vs DSO

	N	Mean Rank	Sum of Ranks
Negative ranks	0[a]	0	0
Positive ranks	32[b]	16.5	528
Ties	0[c]		
Total	32		
Z	-4.937		

a: DSO slower than CGI
b: DSO faster than CGI
c: Equal performance

One-tailed test - DSO faster than CGI

H_0: $\mu_{CGI} = \mu_{DSO}$ H_1: $\mu_{CGI} < \mu_{DSO}$
α: .05 Reject H_0 if $|z_T| > 1.6449$
Conclusion: Reject H_0 as $|-4.937| > 1.6449$ holds

Wilcoxon matched-pairs signed rank test - DSO vs Cached DSO

	N	Mean Rank	Sum of Ranks
Negative ranks	6[a]	10.67	64
Positive ranks	26[b]	17.85	464
Ties	0[c]		
Total	32		
Z	-3.74		

a: Cached DSO slower than DSO
b: Cached DSO faster than DSO
c: Equal performance

One-tailed test - DSO faster than Cached CGI

H_0: $\mu_{\text{Cached DSO}} = \mu_{DSO}$ H_1: $\mu_{\text{Cached DSO}} > \mu_{DSO}$
α: .05 Reject H_0 if $|z_T| > 1.6449$
Conclusion: Reject H_0 as $|-3.74| > 1.6449$ holds

For serving large documents through DSO's on Apache, cached DSO's are recommended for better performance.

3.3.5.1.4 *Summary*

Apache's ability to serve documents from a DSO's cache faster than from disk again shows the advantages of memory-resident data. This theory holds true for all file sizes and it is therefore recommended that caching should be implemented when these kind of application are developed.

3.3.5.2 Information Server

3.3.5.2.1 *Small documents*

Wilcoxon matched-pairs signed rank test - CGI vs DLL

	N	Mean Rank	Sum of Ranks
Negative ranks	0[a]	0	0
Positive ranks	32[b]	16.5	528
Ties	0[c]	a:	DLL slower than CGI
Total	32	b:	DLL faster than CGI
Z	-4.937	c:	Equal performance

One-tailed test - DLL faster than CGI

H_0: $\mu_{CGI} = \mu_{DLL}$ H_1: $\mu_{CGI} < \mu_{DLL}$

α: .05 Reject H_0 if $|z_T| > 1.6449$

Conclusion: Reject H_0 as $|-4.937| > 1.6449$ holds

Wilcoxon matched-pairs signed rank test - DLL vs Cached DLL

	N	Mean Rank	Sum of Ranks
Negative ranks	0[a]	0	0
Positive ranks	32[b]	16.5	528
Ties	0[c]	a:	Cached DLL slower than DLL
Total	32	b:	Cached DLL faster than DLL
Z	-4.937	c:	Equal performance

One-tailed test - DLL faster than CGI

H_0: $\mu_{DLL} = \mu_{Cached\ DLL}$ H_1: $\mu_{DLL} < \mu_{Cached\ DLL}$

α: .05 Reject H_0 if $|z_T| > 1.6449$

Conclusion: Reject H_0 as $|-4.937| > 1.6449$ holds

Cached DLL's were found to perform the best of the different technologies when used to serve small files.

3.3.5.2.2 *Medium-sized documents*

Wilcoxon matched-pairs signed rank test - CGI vs DLL

	N	Mean Rank	Sum of Ranks
Negative ranks	0[a]	0	0
Positive ranks	32[b]	16.5	528
Ties	0[c]		
Total	32		
Z	-4.937		

a: DLL slower than CGI

b: DLL faster than CGI

c: Equal performance

One-tailed test - DLL faster than CGI

H_0: $\mu_{CGI} = \mu_{DLL}$ H_1: $\mu_{CGI} < \mu_{DLL}$

α: .05 Reject H_0 if $|z_T| > 1.6449$

Conclusion: Reject H_0 as $|-4.937| > 1.6449$ holds

Wilcoxon matched-pairs signed rank test - DLL vs Cached DLL

	N	Mean Rank	Sum of Ranks
Negative ranks	0[a]	0	0
Positive ranks	32[b]	16.5	528
Ties	0[c]		
Total	32		
Z	-4.937		

a: Cached DLL slower than DLL

b: Cached DLL faster than DLL

c: Equal performance

One-tailed test - DLL faster than CGI

H_0: $\mu_{DLL} = \mu_{Cached\ DLL}$ H_1: $\mu_{DLL} < \mu_{Cached\ DLL}$

α: .05 Reject H_0 if $|z_T| > 1.6449$

Conclusion: Reject H_0 as $|-4.937| > 1.6449$ holds

Cached DLL outperforms non-cached DLL's when serving medium-sized documents.

3.3.5.2.3 *Large documents*

Wilcoxon matched-pairs signed rank test - CGI vs DLL

	N	Mean Rank	Sum of Ranks
Negative ranks	0[a]	0	0
Positive ranks	32[b]	16.5	528
Ties	0[c]		
Total	32		
Z	-4.937		

a: DLL slower than CGI

b: DLL faster than CGI

c: Equal performance

One-tailed test - DLL faster than CGI

H_0: $\mu_{CGI} = \mu_{DLL}$ H_1: $\mu_{CGI} < \mu_{DLL}$

α: .05 Reject H_0 if $|z_T| > 1.6449$

Conclusion: Reject H_0 as $|-4.937| > 1.6449$ holds

Wilcoxon matched-pairs signed rank test - DLL vs Cached DLL

	N	Mean Rank	Sum of Ranks
Negative ranks	0[a]	0	0
Positive ranks	32[b]	16.5	528
Ties	0[c]		
Total	32		
Z	-4.937		

a: Cached DLL slower than DLL

b: Cached DLL faster than DLL

c: Equal performance

One-tailed test - DLL faster than CGI

H_0: $\mu_{DLL} = \mu_{Cached\ DLL}$ H_1: $\mu_{DLL} < \mu_{Cached\ DLL}$

α: .05 Reject H_0 if $|z_T| > 1.6449$

Conclusion: Reject H_0 as $|-4.937| > 1.6449$ holds

Cached DLL's perform better than non-cached DLL's when serving large documents from Information Server

3.3.5.2.4 *Summary*

Caching seems to have advantages through both architectures as ISAPI DLL's on Information Server also performs best when cached files are served. As with Apache's cached DSO, the cached DLL is recommended as the best implementation.

3.4 FINDINGS

The research hypothesis stated in paragraph 1.2.3 tried to determine if there is a relationship between the performance of a web server and its popularity. Research has shown (NetCraft 2002) that Apache is by far the most popular web server in use on the Internet today. A reformulation of the hypothesis in Chapter 1, one could say:

H_0	:	The popularity of a web server is not related to its performance.
H_1	:	There exists a direct connection between a server's performance and its popularity.

With reference to the test results of Chapter 3, it was found that Apache performed better than Information Server in six different tests, with Information Server being faster in only two. Four tests performed found that neither of the two servers were faster than the other. Based on these results, we can say with a 95% certainty, overall, that Apache performs better than Information Server over various load levels.

	Small	Medium	Large
Static	-	Apache	Apache
CGI	Apache	Apache	-
DSO	-	Apache	Information Server
Cached DSO	-	Apache	Information Server

Table 3.2 Web server performance comparison summary

When choosing a web server platform, many factors should be considered. These could be technical expertise, stability and even some obscure and archaic corporate policy. One should however never loose sight of what is really important to the application of the technology and the client's interest. When assigning funding to any project, ROI justification will become more important in the current economic environment, placing more strain on all business units, including Information Technology. Increasing the value of content and services through improved usability and reduced cost per transaction leads to more efficient and quicker businesses and business processes (Oglethorpe 2002). It makes business sense to be able to get more for the same expense. It is therefore found that popularity is related to performance, hence a rejection of the null hypothesis and acceptance of the alternative hypothesis. An important factor that has to be kept in mind at all times is that server performance is application specific. It is recommended therefore that when technologies are chosen, the best technology for the specific application has to be selected.

3.5 SUMMARY

Measurement is a tool that can be applied to software engineering to deliver better, faster and more stable products. It has the following advantages for any software project:

- It will ensure the objective comparison of different technologies;
- It will identify good and poor characteristics of systems;
- It will provide confidence to the project team that the technology applied will perform within a set range of parameters;
- It forces the team to compile a thorough list of requirements they set for the system or technologies, which will be useful in the ongoing evaluation of the technology and will provide a guideline by which requirements will be met with the project's completion.

The tests conducted in this chapter support decision-making efforts in the planning of Internet-based software. This information should be used to select a server platform that matches the application profile of the system. Chapter 4 describes the efforts in building a multifunction web server for both server platforms and testing the results obtained here with an actual Internet system. Using similar technologies and approaches on the two systems will ensure parallel testing of the servers' characteristics.

CHAPTER 4

THE APPLICATION OF DYNAMIC DEVELOPMENT METHODOLOGIES

4.1 INTRODUCTORY REMARKS

When a company embarks on a web development venture, many different technologies and methods have to be evaluated to choose the correct one. As web development is commonly seen as an advanced version of Desktop Publishing (DTP), managed by highly skilled server engineers, different opinions exist around the science of Internet development. In order to partake in a successful Internet venture, parties involved have to understand all areas covered. Apart from knowing how to create HTML pages and uploading them to web servers, the developer (or designer) has to know the differences between operating systems and web servers, their technological strengths and weaknesses.

Due to the nature of the Internet and its development path, two main areas of expertise exist, being Internet design and Internet development. Internet design is mainly focussed on the graphical representation of content and media on the Internet. This is the next level of DTP and is mainly concerned with the creation of static content to be placed on web servers. Internet development is the technical side of dynamic content provision on the Internet. It focusses on building interfaces to dynamic data sources (like databases) and the development of dynamic content generators such as online charts and search engines. To make informed decisions on using specific Internet technologies, management and project teams should understand this thoroughly. Figure 4.1 shows a breakdown of the methodologies.

At this point the focus will move to the dynamic section of this development structure. Creating dynamic content on Internet servers was traditionally only possible to Internet programmers. It is usually done through one of two methods, namely scripting and server extension.

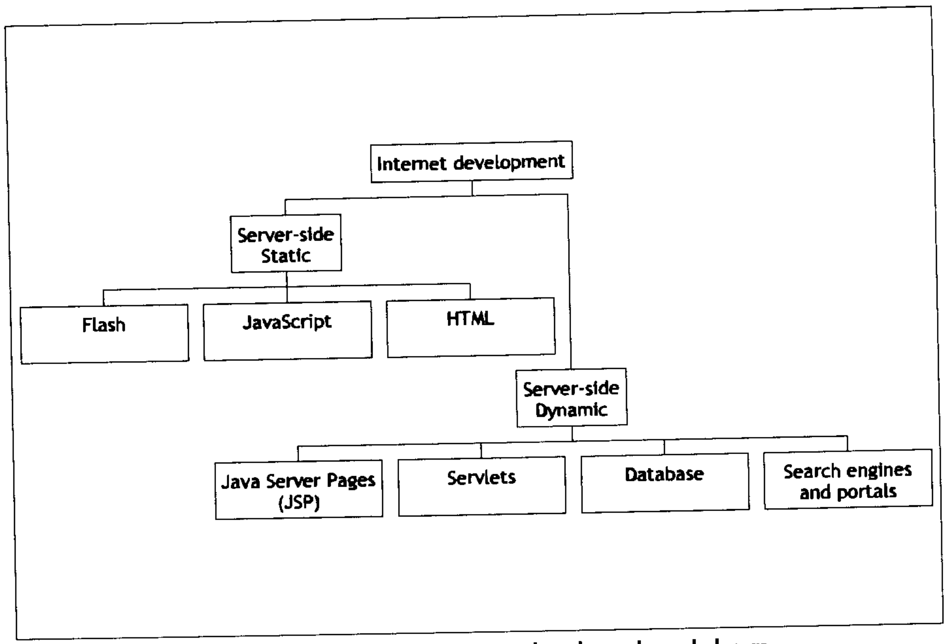


Figure 4.1 Internet development technology breakdown

4.2 DIFFERENTIATING TECHNOLOGIES

The technologies of scripting and server extension are related, but operate in totally separate ways.

4.2.1 Scripting

There exist two different methods of deploying scripting in terms of Internet development: Client-side scripting and server-side scripting. Client-side scripting is executed inside the user's browser. The best known implementation of this is probably JavaScript and all its different mutations and versions. Client-side scripting requires no processing from the server and is client-only. Client-side scripting can not interface with databases and data stores on the server. A JavaScript runs inside of a web browser that supports JavaScript (HTMLGoodies.com, 2002). It is usually applied for aesthetical reasons or for the provision of limited client-side interactivity, but not for interaction with the server.

Server-side scripting has more functionality for server interaction. If the scripting engine supports it, a server-side script can interact with databases, locally or remotely. It has limited support for client interaction and is limited to data sent with the client's request to the server. Scripting languages require an engine to run on. Without it, it is the same as a normal HTML page. The engine parses the page, like interpreters run programs, compiles the final output and send it to the web server which serves it back to the client. Figure 4.2 shows the close comparison between interpreters and scripting engines.

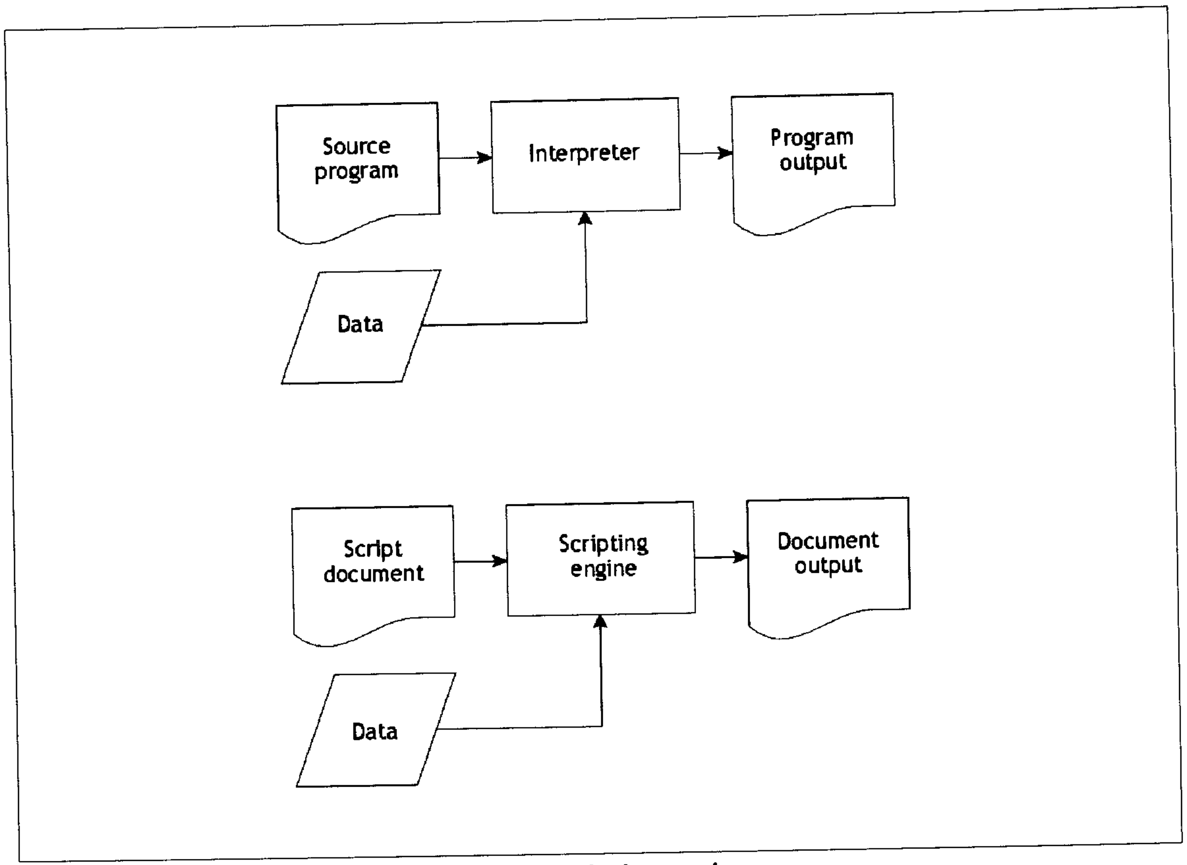


Figure 4.2 Comparing interpreters to scripting engines

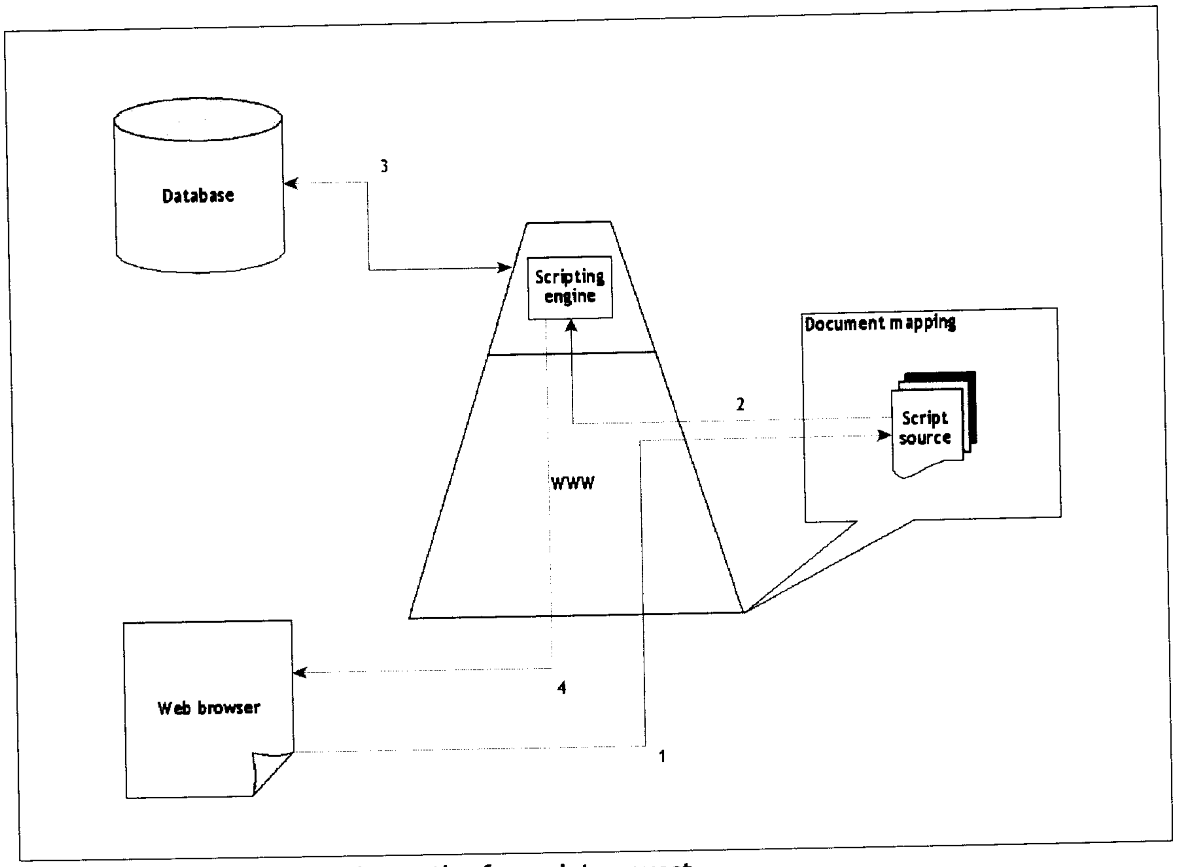


Figure 4.3 The execution path of a script request

Figure 4.3 shows the execution path of script-based web server requests.

A request to a script-based document is handled as follows:

- The web browser connects to the web server and requests the script file;
- The server loads the file from its storage system and compares the document to its registry of file mappings. The web server feeds the document, together with the request data to the scripting engine;
- The engine does its file manipulation and database interaction based on the document's contents; and
- The final document is sent back to the web server which gets passed back to the client.

The most common of these server-side scripting technologies would be Microsoft's Active Server Pages (ASP) and the open-source Perl. ASP's are only supported on Microsoft web servers, while Perl is supported across platforms.

4.2.2 Server extension

Server extensions are created in a number of different ways. Two of the most popular of these methods are ISAPI (Internet Information Server) and DSO (Apache). After being loaded, a server extension runs as part of the core web server. As being part of the web server, a server extension has access to server-side databases and files, which makes it excellent for developing interactive Internet systems. Server extensions do not use input files like scripts. All inputs and outputs of the extension are handled and retrieved from inside the extension. This means that there is less interaction between server components than with server-side scripting. Figure 4.4 displays the execution model of requests passed to web server extensions.

Requesting a document through a server extension is different from the script-driven system. It has fewer steps than the scripting model and goes through the following phases:

- The web browser connects to the server and requests the document directly from the extension;
- The extension interacts with the database and other data sources and builds the output for the request; and
- The final document is then sent to the client.

Where server-side scripting compares to interpreters, server extensions compare to compiled executables (Figure 4.5). This suggests that server extensions will have the same benefits as what compiled applications have.

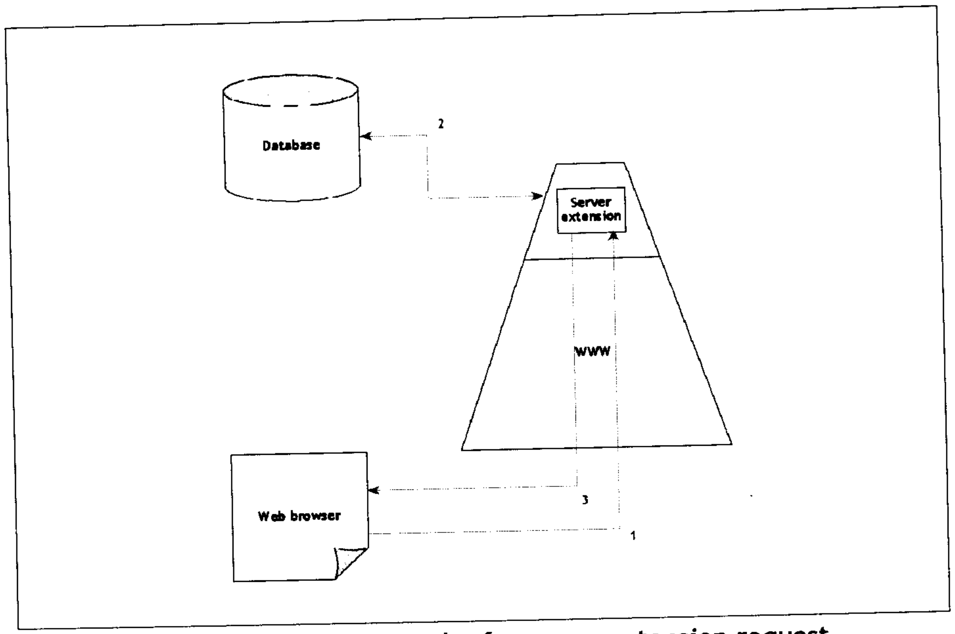


Figure 4.4 The execution path of a server extension request

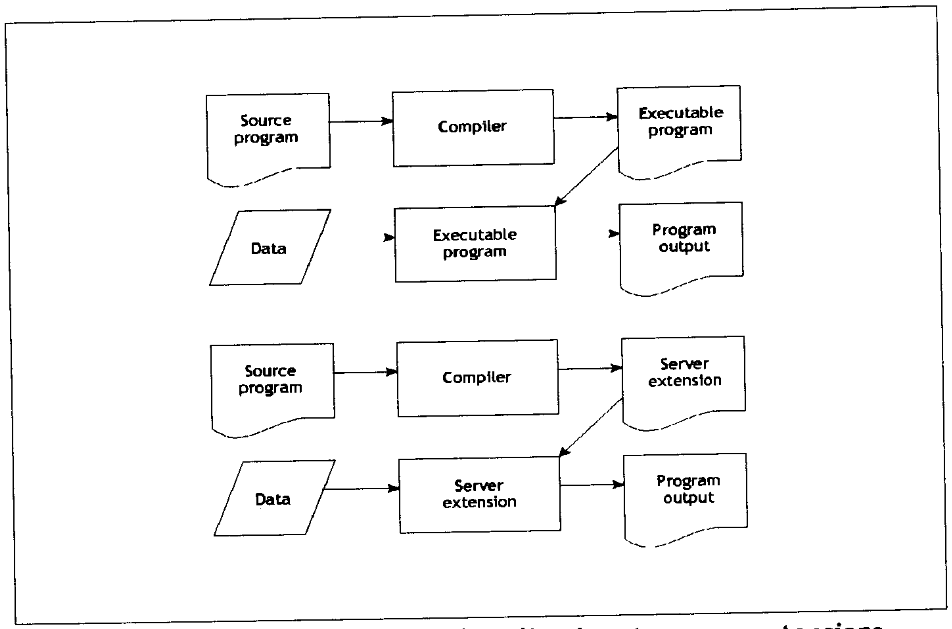


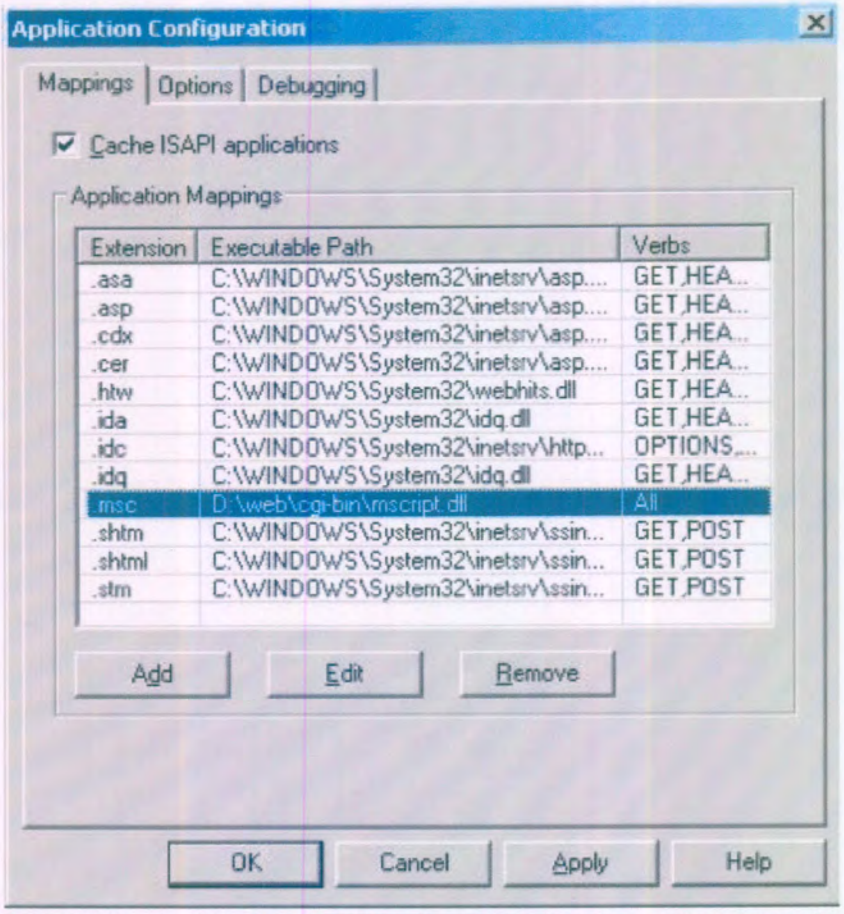
Figure 4.5 Comparing compiled applications to server extensions

4.3 CONSTRUCTING A TESTING FRAMEWORK

As both servers support server extensions as well as scripting, a dynamic server environment was created. A company information portal was developed to interact with a sample human resource database.

Microsoft's ASP model is based on server extensions. ASP is the premier scripting technology used on Microsoft servers today. As a result, it was decided to develop a similar scripting language and engine to interface with the database. Using the performance results obtained in the previous chapter, it was developed using Dynamic Shared Objects (DSO) for Apache/Linux and Internet Server API (ISAPI) for Internet Information Server/Windows.

A set of document outlines were created for deployment on both servers. The documents are requested from the servers in two different ways, one being through a server extension and the other though its filename which is mapped to a specific server module. Figures 4.6-4.8 show the mapping procedures to map documents to DSO and DLL handlers.

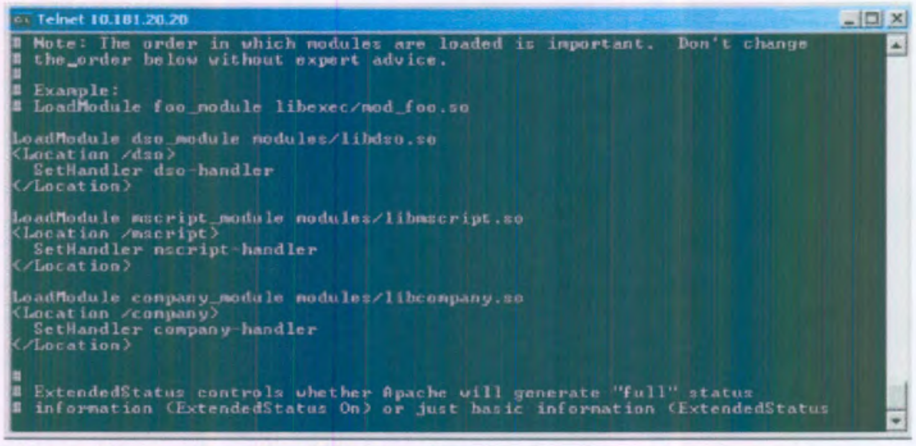


☑ Cache ISAPI applications

Application Mappings

Extension	Executable Path	Verbs
.asa	C:\WINDOWS\System32\inetsrv\asp....	GET,HEA...
.asp	C:\WINDOWS\System32\inetsrv\asp....	GET,HEA...
.cdx	C:\WINDOWS\System32\inetsrv\asp....	GET,HEA...
.cer	C:\WINDOWS\System32\inetsrv\asp....	GET,HEA...
.htw	C:\WINDOWS\System32\webhits.dll	GET,HEA...
.ida	C:\WINDOWS\System32\idq.dll	GET,HEA...
.idc	C:\WINDOWS\System32\inetsrv\http...	OPTIONS,...
.idq	C:\WINDOWS\System32\idq.dll	GET,HEA...
.msc	D:\web\cgi-bin\mscript.dll	All
.shtm	C:\WINDOWS\System32\inetsrv\ssin...	GET,POST
.shtml	C:\WINDOWS\System32\inetsrv\ssin...	GET,POST
.stm	C:\WINDOWS\System32\inetsrv\ssin...	GET,POST

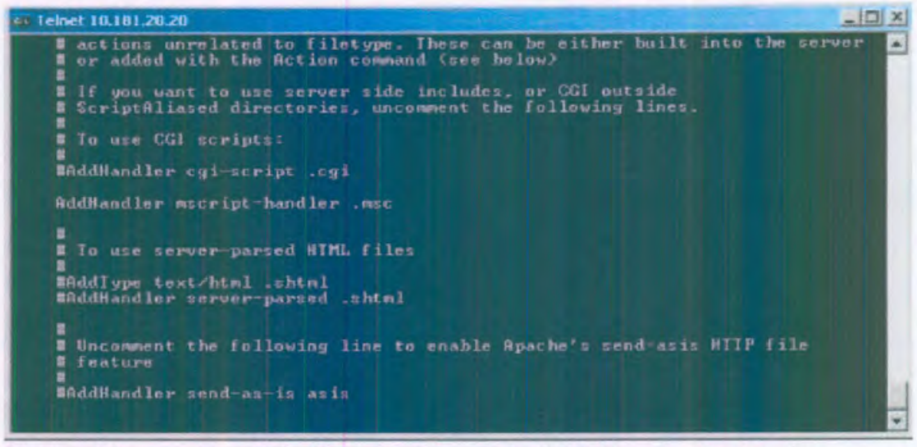
Figure 4.6 Mapping scripting documents to server extensions with Internet Information Server



```
# Note: The order in which modules are loaded is important.  Don't change
# the order below without expert advice.
#
# Example:
# LoadModule foo_module libexec/mod_foo.so

LoadModule dso_module modules/libdso.so
<Location /dso>
  SetHandler dso-handler
</Location>

LoadModule mscript_module modules/libmscript.so
<Location /mscript>
  SetHandler mscript-handler
</Location>

LoadModule company_module modules/libcompany.so
<Location /company>
  SetHandler company-handler
</Location>

#
# ExtendedStatus controls whether Apache will generate "full" status
# information (ExtendedStatus On) or just basic information (ExtendedStatus
```

Figure 4.7 Creating DSO mappings for Apache

When a mapped document is called by a client, it is fed to the document handler (scripting engine) by the web server, parsed and served to the client. The technology is the same between the two web servers, which makes them worthy competitors.



```
# actions unrelated to filetype. These can be either built into the server
# or added with the Action command (see below)
#
# If you want to use server side includes, or CGI outside
# ScriptAliased directories, uncomment the following lines.
#
# To use CGI scripts:
#
#AddHandler cgi-script .cgi

AddHandler mscript-handler .msc

#
# To use server-parsed HTML files
#
#AddType text/html .shtml
#AddHandler server-parsed .shtml

#
# Uncomment the following line to enable Apache's send-asis HTTP file
# feature
#
#AddHandler send-as-is asis
```

Figure 4.8 Mapping document types to DSO libraries in Apache

In addition to the scripting framework developed, the same document handlers were created as part of the server extension in a DSO and ISAPI dynamic module. The difference between the two is that with the scripting, the files exist as text files on the web server and serves as input to the extension, rather than being a part of the extension. By definition, one would expect a slight performance difference between the two technologies, given the extra processes required to execute the script request.

The testing process for this chapter is aimed at determining the best performing technology on each server. The development of the server extensions were done on two Borland products, namely Delphi Enterprise 6 (Windows-based) and Kylix Server 1 (Linux-based). These two products were chosen for various reasons:

- Both solutions support Object Pascal;
- The source code is identical between the two;
- Their components and technology base overlap;
- Delphi has proven itself as a high performance RAD development tool for enterprise and Internet solutions; and
- Kylix has been voted the best development tool for Linux (Borland, 2002).

To eliminate performance issues specific to the database system, a cross-platform database was selected. Borland's InterBase 6 has proven itself as a high performance relational database management system. Initially developed for the UNIX platform, InterBase was created with a multi-thread architecture in mind. InterBase is currently supported on Solaris, Microsoft and Linux platforms, with a huge following after the database was temporarily released as an open-source product. The main reason for choosing InterBase for this test is its ability to be deployed on both platforms used here.

One problem unfortunately caused some serious implications for a cross-platform performance comparison. The database connectivity support for Kylix is currently inferior to the offerings on the Microsoft platform. This was decided based on the performance comparison to single platforms only. This is implied by the performance results obtained with these tests. Tight clustering can be noticed on the combined performance charts for Apache (Figure 4.9). This suggests a serious bottleneck within the architecture of the Apache-based solution.

4.4 PERFORMANCE DATA

4.4.1 General remarks

Figure 4.9 depicts Apache's performance, with Figure 4.10 Information Server's performance data.

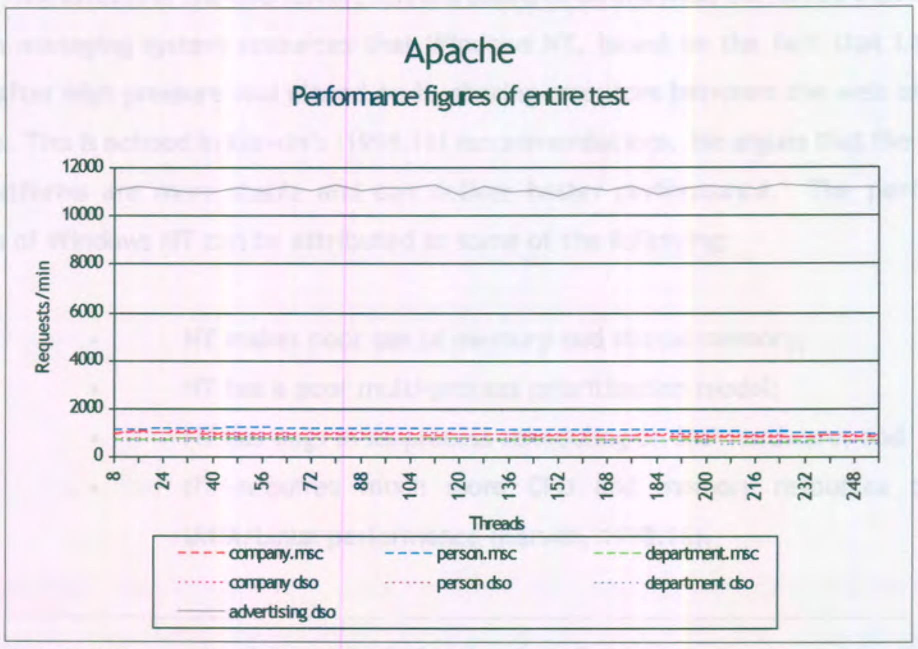


Figure 4.9 Combined Apache performance chart

The first apparent difference between these two charts is the high level of clustering experienced on the Apache tests, as opposed to the dispersed series of Information Server. Both servers were tested with six database-driven pages and one non-database control test. The 'advertising' series serves a random image from memory to the client. This series is used as a benchmark for standardisation purposes[1].

Apart from the problem mentioned with regards to the database interface components for Kylix, another interesting observation was made. The InterBase performance recommendations state that if InterBase is to be used as a data store for Internet systems, the database and web

[1] For all charts, the legend labels for scripting ends in '.msc' while DSO requests end in 'dso'

server should be separated and placed on two different machines when used on Windows (Karwin, 1998:15). This was confirmed in the IIS performance tests when a single machine was used to run the web server and database. When the server was placed under too much stress, the entire machine froze up. After splitting the two servers, very high output figures were obtained on Windows. When the same was tried on Linux, the opposite was experienced. Significantly lower performance was measured and the two servers were once again combined. However, the effects of the two servers running together on one machine hinted that Linux was better in managing system resources that Windows NT, based on the fact that Linux kept running after high pressure was placed on it, sharing resources between the web server and database. This is echoed in Karwin's (1998:16) recommendations. He argues that the UNIX and Linux platforms are more stable and can deliver better performance. The performance problems of Windows NT can be attributed to some of the following:

- NT makes poor use of memory and virtual memory;
- NT has a poor multi-process prioritization model;
- NT has bugs in its process scheduling on SMP hardware; and
- NT requires much more CPU and memory resources to match UNIX/Linux performance (Karwin, 1998:16).

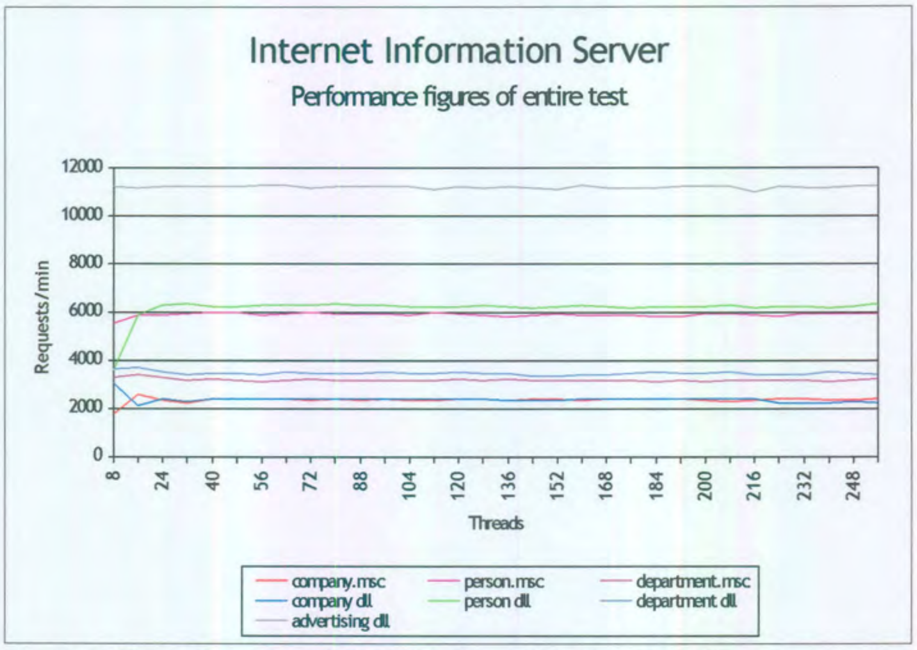


Figure 4.10 Combined Internet Information Server performance profile

Given that the performance band of Apache/Linux seems to be around the 1000-mark, regardless of content, it would be unfair to assume that Information Server has the upper hand in serving test clients at high stress levels to Apache. As the development of these kind of applications, using a new development environment and components, is still in its infancy when compared to the legacy already created by thousands of Windows developers, the statement is being made that the main bottleneck in this system could be the interface to the database. This is further accentuated by the fact that the two servers are running identical databases by the same vendor, utilising the same database engine version, only on different platforms. Due to a lack of equally powerful components for Kylix, it was decided to split the performance comparison between the servers and focus solely on the different tests on the same server.

These tests focus on the performance differences between server extensions and the scripting engine developed for the test. By comparing the performance figures between the technologies, it will give the reader a better understanding of the technology and its implications on business and the system's architecture.

4.4.2 Apache

Figure 4.11 shows the different performance profiles for the database-driven pages. Each of these pages contained different content and used different database tables to retrieve data from. From this chart it can be seen that in two of the three cases more script-based documents were served than extension-driven pages. Whether this is caused by the interface component shortcomings, would be hard to say. It was therefore found that on Apache script-based solutions could perform better than extension-based systems.

These findings are reenforced by the popularity of scripting with Apache. Perl, which is incidently also cross-platform, had over 1,000,000 users in 1999 (www.perl.org, 2002).

The close clustering in the chart reiterates the concern that there exists some kind of bottleneck in the interface between the database and the web server extensions, varying in total numbers between 600 and 1300 requests. Another observation that can be made from these performance charts is that all the performance charts tend to deteriorate slightly as the number of clients and requests increase.

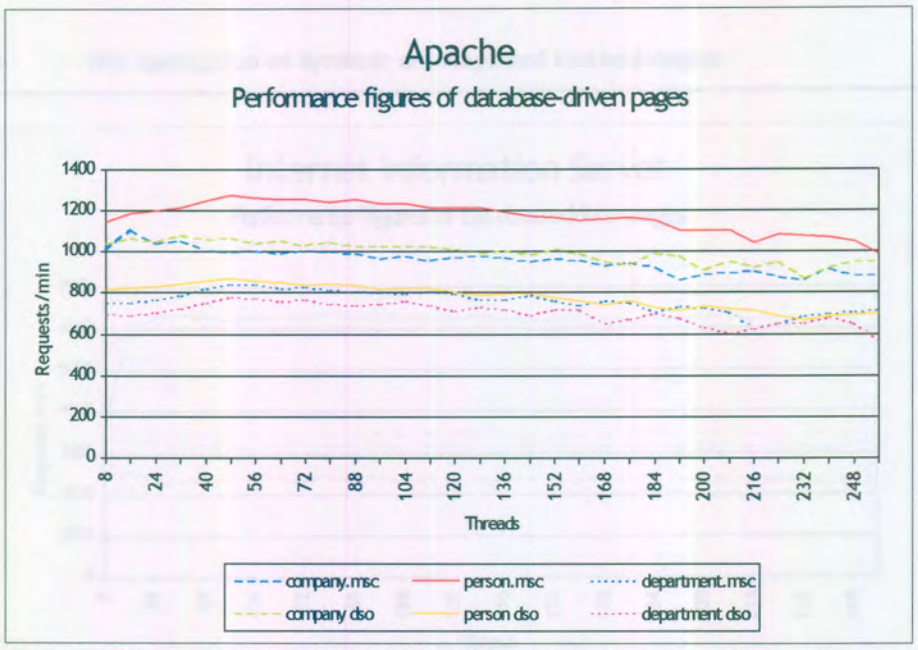


Figure 4.11 Apache: Performance figures of database-driven pages

4.4.3 Internet Information Server

In contrast with the performance figures obtained from the Apache tests, the performance charts for the Information Server show three different performance clusters (Figure 4.12), grouped per test.

The top pair is the 'person' page. It contains the least amount of information and has two database queries embedded within. The second pair (department) contains more output data than 'person' and also contains two different database queries. Finally, the 'company' page contains only one database query, but has the most data output. In two of the cases, the server extensions outperform the scripted version of the file. It is therefore strange that given these results, that scripting has become so popular in current web development projects on Windows NT, especially ASP.

The conclusion is therefore made that within the Windows NT environment server extension should be considered to scripting as the preferred development technique for Internet systems.

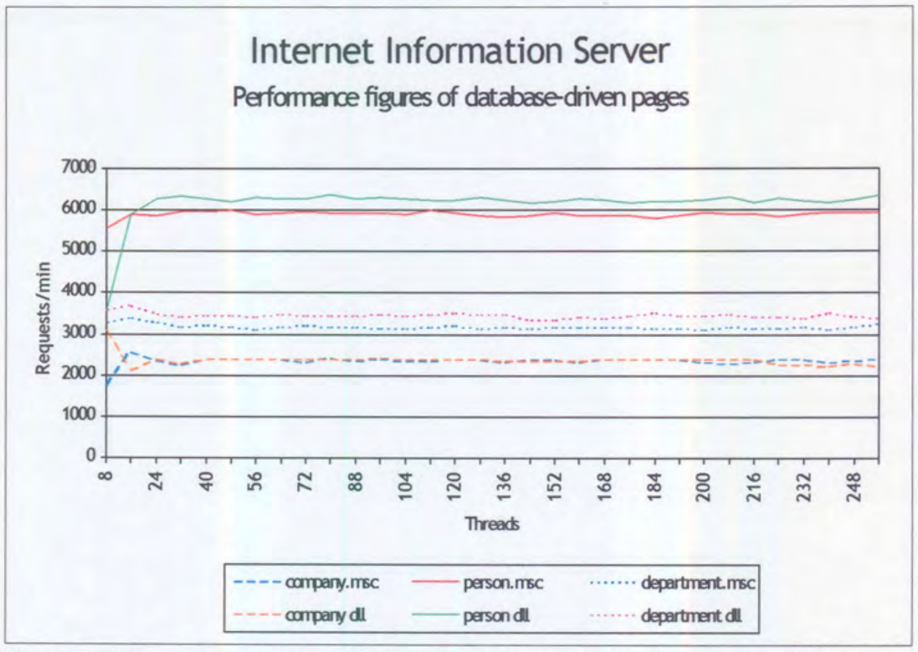


Figure 4.12 Internet Information Server: Performance figures of database-driven pages

4.4.4 Summary

The performance test for this real world application of server extensions and scripting for Information Server and Apache suggests that Information Server is definitely geared towards the use of server extensions rather than scripting. Apache, on the other hand, is optimised for the deployment of script-based solutions. Due to the nature of script parsing it is argued that Linux has advanced capabilities for text manipulation. This in not surprising due to the fact that most Linux applications, services and the operating system itself is highly dependable on text files for configuration purposes.

4.5 ARGUMENTS AROUND SCRIPTING AND SERVER EXTENSION

Many arguments exist around these two methodologies for Internet developers. To create an objective view of the different approaches, the characteristics of these two methodologies will be compared.

4.5.1 Characteristics of scripting

Advantages

- Scripting languages can be relatively easy to learn;
- Files containing scripts can be changed with a text-editor, as they are all text-based;
- These files are relatively small and easy to understand;
- Development is quicker;
- The run-code is safer, as errors are to be handled by the scripting engine, not the program (script) itself;
- Better security can be forced down on the server with a single control mechanism;
- Smaller code and files;
- Scripts can easily interact with database systems;
- It can be developed in many different languages;
- Some languages support advanced data structures like object oriented programming;
- Many deployments do not need any complicated development environments or external engines. Many web servers are distributed with the engine included; and
- Advanced processes, like memory management is handled by the scripting engine and not by the script.

Disadvantages

- Scripting languages can have limited functionality;
- Scripts are interpreted (parsed), which tends to slow them down and can therefore be inefficient;
- A lack of a run-time compiler can allow errors to occur during execution;
- Some scripting languages are only supported on one or two platforms, e.g. ASP;
- Scripting languages are typeless and rely on more complex components to support its operation (Ousterhout, 1998:1); and
- Few proper integrated development environments (IDE) exist for script-development.

4.5.2 Characteristics of server extensions

Advantages

- High complexity can be achieved through server extension;
- Server extension can be much faster as the extensions are already compiled and do not need any interpretation;
- Error checking can be performed before deployment;
- Strong typing enforces better control over program execution;
- Many IDE's exist for the development of server extensions;
- Server extensions can be developed in multiple languages like C++ and Object Pascal;
- More efficient database interfacing can be achieved;
- Server extensions are much more stable than scripts; and
- It can provide much more complex interfacing between disjunct systems.

Disadvantages

- The skill levels required to build server extensions are much higher than with scripting;
- To change server extensions, an IDE and compiler are required;
- Server extension files tend to be larger than scripts, delaying deployment;
- Advanced operations, like memory management and object manipulation have to be handled in the server extension; and
- Development can take more time than script-based methodologies.

4.6 FINAL CONCLUSIONS AND ARCHITECTURE RECOMMENDATION

There are advocates for both of these technologies and each of these has his own reason for that. However, the task at hand is to do an objective evaluation of what is available and highlight the differences, advantages and disadvantages of technological issues. This section will highlight the motivations behind the different technologies when evaluated for corporate deployment.

4.6.1 Scripting

Scripting is currently very popular for web development. It requires relatively average development skills as many developers using this methodology need to know little more than HTML coding to create script-driven web sites. Development compares to the model of rapid application development (RAD) and is usually seen to be on a higher language level (Ousterhout, 1998:1). Deployment of these files is also quite easy, provided access to the server or storage-device is simple and quick. Script-based technologies can eventually be seen as an extension of current desktop publishing activities. Scripting languages also have financial advantages. Many scripting engines are distributed free of charge, or at a very low cost. The ASP scripting engine, for example, is distributed with Information Server and is currently actively supported by Microsoft. There are also little expenses as far as development areas are concerned, as most, if not all, of these documents can be edited in the most basic of text editors. Script-based systems also have the ability to create quite complex structures and algorithms which makes it very versatile.

Security of the system can be centrally managed by the server's administrator. Care should however be taken in choosing a script-based technology. Scripting engines can be resource intensive and may need highly trained administrators to configure and maintain the servers. Furthermore, if the system is to be used for commercial purposes, it should be noted that transactions could be executed at a lower speed due to the interpreted nature of script-driven systems. Some web servers might not support the desired technology and may only be available to some operating systems or web server platforms. Incorrect configurations of servers can also leave the server and documents residing on the machine open to security exploits which can cause havoc to the server, the company's financial position and its reputation.

Script-based solutions should be chosen under the following circumstances:

- High volumes of documents are produced for web distribution, e.g. portal services;
- The technical expectations of the system component are relatively simple and do not require advanced techniques;
- Performance of the system is non-critical to its use and application;
- The level of technical expertise is centred around system administrators and engineers and there is a lack of formal programming

expertise;

- The planned system is not to play the central role of the company's business model, e.g. e-commerce and banking; and
- The company is not able to afford expensive development tools to create server extensions, e.g. Borland Delphi, Microsoft Visual Studio, etc.

4.6.2 Server extension

First and foremost, server extension through 4th generation languages and development environments like Delphi, Kylix and C#, gives the developer high levels of control and empower him to develop highly complex systems with virtually unlimited functionality. With the source code of server extensions being compiled by a language compiler, better error checking is performed and higher quality is guaranteed. Server extensions provide better stability and performance and can be used for mission-critical applications, such as e-commerce systems. Better resource utilisation is obtained, both on the server and other remote data sources. Server extensions can be much more efficient than script-based solutions.

Server extensions do however require better skilled developers to develop it. They have to be familiar with formal development methodologies and techniques and mostly need to understand the technicalities of the underlying systems and structures. Server extensions also require expensive development tools, such as Delphi and Visual Studio. Many web servers support only a handful of server extension interfaces and might require server- and operating system specific development environments. As the developer has access to many advanced features of the operating system, he is responsible for the quality of the code. Development of these extensions can take much longer than script-based solutions due to the inherent complexity of the technology. Maintenance of existing systems is also much more complex than script-based systems. Lastly, highly skilled system engineers might be required to configure servers for these extensions.

Server extensions could be considered if the situation in which it has to be deployed, matches the following criteria:

- Highly complex methods are to be incorporated into the system;

- The system will perform a mission-critical function in the company and is used for commercial application;
- Performance and reliability are highly important to the success of the project;
- The system will combine the data and functionality of several remote and complex data sources and systems;
- There is sufficient expertise available for the project in both development and support areas; and
- The development team has access to advanced development tools to build server extensions.

4.7 CLOSING REMARKS

It will be a futile exercise to force any development team to use any specific technology without considering its implications, requirements, abilities, architecture and limitations. By objective analysis, the developer can create a profile of the company and the project and come to a formal decision on the technology deployment. Each project has to be evaluated in terms of its inputs, outputs, processes, data sources and -requirements and risk factors.

Sufficient analysis into the project and its environment will yield good results for the project as well as the company. However, to make a good system better, evaluation should be performed of the system and its interaction with its environment, users and data sources. Good design and development principles are not enough to raise the bar for software development excellence. More is needed. A careful look into system analysis and development techniques might yield better results, combined with a thorough understanding of the technical and business aspects and impact of technologies involved with Internet and e-commerce development.

CHAPTER 5

CLOSING REMARKS AND RECOMMENDATIONS

5.1 INTRODUCTORY REMARKS

Performance testing is a good tool to preempt a system's expected performance, or to determine the best technologies to apply. In this research, it was used to decide which web server to use for specific server applications. Should one look at the Systems Development Life Cycle (SDLC), Whitten, Bentley & Barlow argues that the design-phase is sometimes used to decide on the technological aspects of a project (1994:112). The first section of the design-phase defines a design target. Information for this section is based on three sources (Whitten et al, 1994:112-114):

<u>Design ideas and opinions (various sources):</u>

This is a cauldron of ideas and bits of knowledge thrown together to create a Cloud of Creativity;

<u>Business requirements statement (systems analysis-phase):</u>

This includes technical and operational requirements from business and encapsulates the essence of the project or system; and

<u>Approved technology architecture (system managers):</u>

Many companies have formal specifications for approved and tested software systems. If any system is used to support development or operation of deployed systems, it can be used with few problems and little bureaucracy. If systems are not approved outright, this creates an opportunity.

When systems are not authorised by the appropriate system managers, it might be worth while to investigate the reasoning behind it. If the system has not been evaluated at all, or the evaluation of the system has been done a long time ago on older technologies, include it in the evaluation stage again. The model by Whitten et al (1994:113) extends to Design phase 2A (Figure 5.1). This involves the acquisition of the necessary hardware and software. When carefully considered, the performance tests performed previously in this study should be included in this stage.

Whitten et al (1994:115) argues that the acquisition phase is not a common element of most SDLC's and methodologies, as many people believe that "we will build what we need". One can accept that this commonly excludes software elements like operating systems, database servers and web servers. Nonetheless, the developer should never overlook the thorough and objective evaluation of all elements chosen for a system's deployment. The model states that this phase is centred around the communication of system requirements to the vendors followed by the usual Requests for Proposals and quotations. It is however important that each system evaluated or considered for application in the system be evaluated by the development team based on the application and environment of the system. Even though the preceding tests in this study suggest that the faster system should be the better for dynamic deployment, it should be tested within the environment in which it will operate. Furthermore, one has to realise that the speed of a system might not be the only driving force behind its selection. There are far more elements and factors working in on a system, its operation and its success than what meets the eye. Deploy fit-for-purpose technologies to obtain best results.

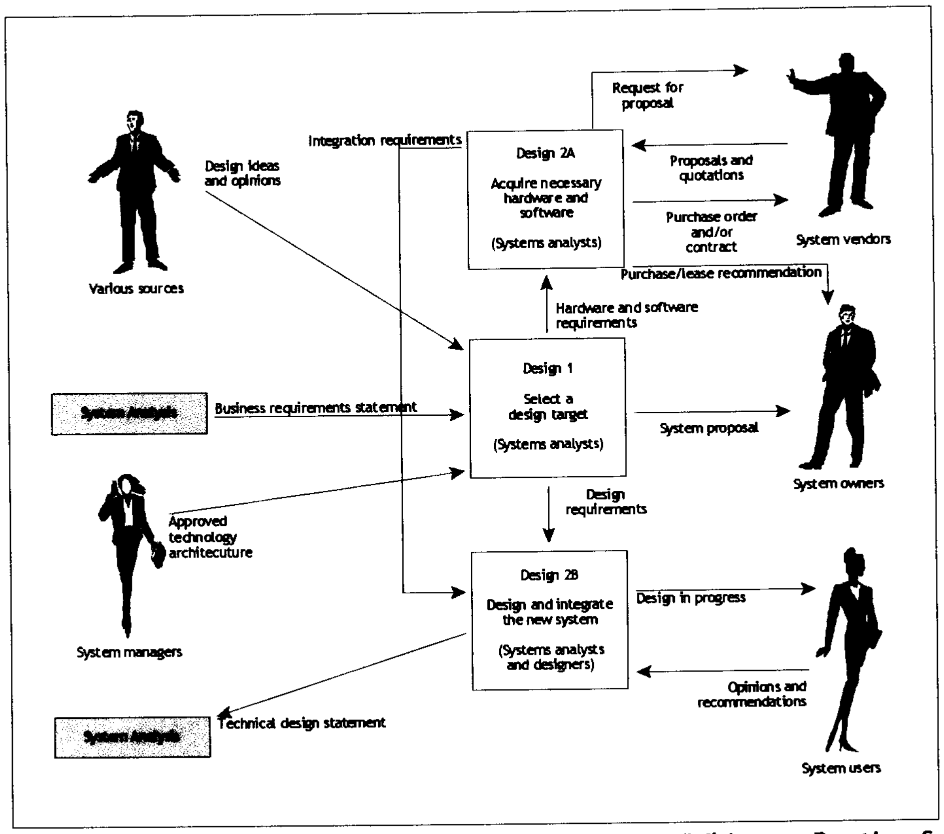

Request for proposal
Integration requirements
Design ideas and opinions
Design 2A
Acquire necessary hardware and software
(Systems analysts)
Proposals and quotations
Purchase order and/or contract
System vendors
Various sources
Purchase/lease recommendation
Hardware and software requirements
System Analysts
Business requirements statement
Design 1
Select a design target
(Systems analysts)
System proposal
System owners
Approved technology architecuture
Design requirements
Design 2B
Design and integrate the new system
(Systems analysts and designers)
Design in progress
System managers
Opinions and recommendations
System Analysts
Technical design statement
System users

Figure 5.1 System design phases of the SDLC (Whitten, Bentley & Barlow, 1994:113)

Ahituv, Neuman & Riley (1994:100) define open and closed systems. Closed systems have no environment. There is no interaction between its components and elements outside of its own capsule. Open systems, on the other hand, interact with their environment, similar to most business systems. This also holds true for a software engineering project. If we compare a software development team to a company, Marx, Rademeyer & Reynders (1991:43) lists eight components to the "organization": Entrepreneurship, technological knowledge and abilities, human resources, finances, customer requirements, internal abilities, competition and natural resources.

Most of these elements are external to the project by nature. When compared to the development process it defines it as an open system that is vulnerable to external factors. This holds for the software system as well, especially if this system will be used for business and commercial use. If the ultimate goal of a software project is success, one has to decide what that encompasses. In the general application of systems as business tools, it will invariably be defined as an income generator or at least a complement to the business and its operations. Inherently, all software development processes run the risk of failing, exposing business to take risks on manpower, time and money.

In order to adopt an adaptive business- or process model, the developer has to look at the different forms a system can take on. A system can therefore be open, closed or somewhere in between. Each of these incarnations has its own set of characteristics. Charles Seashore defined three different levels of Group Functioning (Mink, Shultz, Mink, 1986:51). Software projects always involve people and groups of people. Seashore's model can be applied to the elements of a software development project. By improving the essence of a project team, the final output can be improved.

	Level 1	Level 2	Level 3
What strategy is the group following?	Development of collusion.	Confrontation	Collaboration
What is the group seeking?	Confirmation of self	Surprise	Synergy
What is the goal if the strategy is followed?	Security	Some dissonance	Inquiry and search
What learning is possible?	None	Some	Great
What is the state of the system?	Closed	Partially open	Open

Table 5.1 Seashore's levels of Group Functioning (Mink, Schultz, Mink, 1979:51)

From the properties of systems, as described by Seashore, the benefit of open systems hugely outweighs that of closed and partially closed systems. Unfortunately, the developer has to realise that the more open a system is, the more difficult it becomes to maintain and control. This is due to the higher number of external influences on the system.

5.1.1 System states

In order to understand the different system states and their interactions with the external environments, the three states are discussed.

5.1.1.1 <u>Closed system</u>

In its purest form, closed systems exist only in theory (Figure 5.2). These systems have no interaction with their external environments. They are self-sufficient and have no outputs. There are no real-life examples of these conceptual systems.

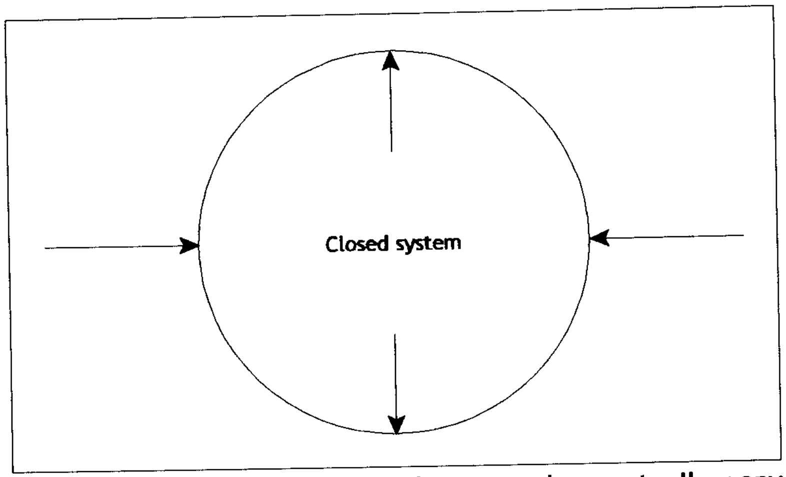


Figure 5.2 A totally closed system does not allow any inputs to enter it, and no outputs to leave it

5.1.1.2 Partially open system

A partially open system allows limited interaction between itself and the environment (Figure 5.3). This ensures control over the system and its environment, but it could debilitate the system's ultimate goal and processes. These systems are also much slower to adapt to change and are usually conservative.

Access to and from the system is controlled by a gatekeeper. The gatekeeper controls access to channels of communication or to particular resources or procedures within a system and its environment (Carter, Martin, Mayblin, Munday, 1988:111). Gatekeepers can have a serious impact on the system and its performance, as they control all cross-border interaction and might have direct relationships with important modules or members of the system.

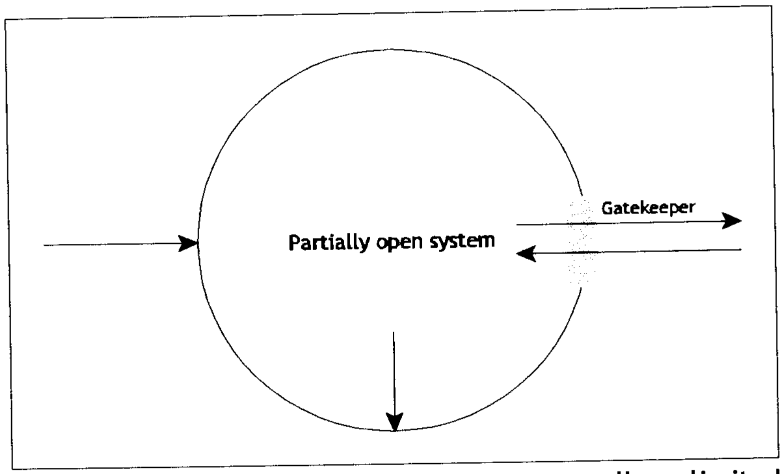


Figure 5.3 A partially open system allows limited interaction between it and its environment

5.1.1.3 Open system

A system that is completely open has unlimited interaction within itself and between its components and its environment. It gets stimulated by almost any external impulse. It can exchange energy, information or materials. The system merges into its environment, up to a point that it has no arbitrary boundary or stable identity and is hard to manage, control and plan for (Carter et al, 1988:7).

Ahituv, Neuman & Riley (1994:100) state that organisations and information systems are open systems. As project teams are subsystems within their organisations, it is based on the same principles. A system that responds to external stimuli as it happens is difficult to control. A tool is required to manage the interaction and to provide a membrane to seal the system into a controllable entity. Something is required to work in conjunction with current methods and structures to reduce the negative influences from entering the system which could cause high risk situations within the system and its processes (See figure 5.4).

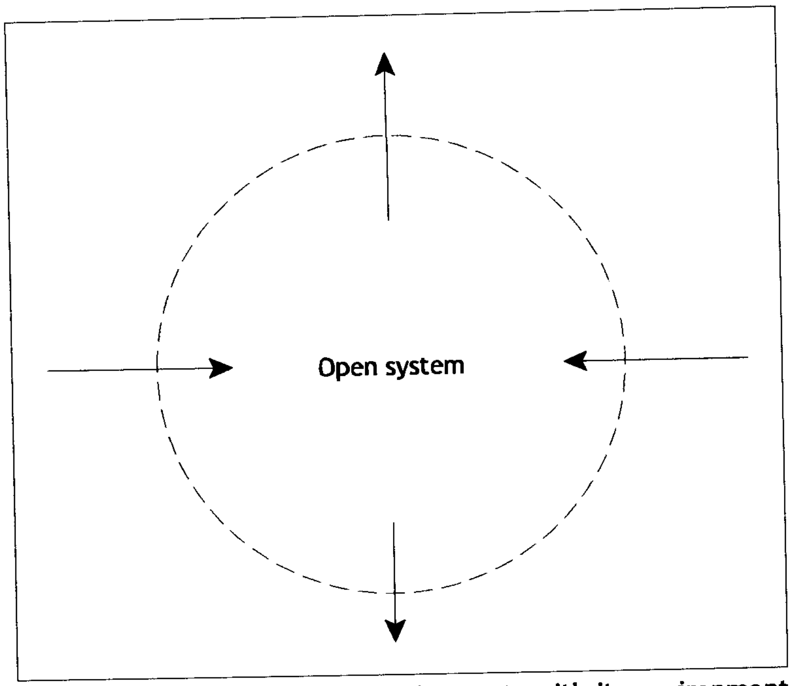


Figure 5.4 An open system interacts with its environment exchanging energy, information and materials

5.2 RISK MANAGEMENT

Risk management is an all-purpose process of controlling uncertain outcomes. It is used to evaluate the current environment and its elements and controlling the impact certain events can have on our lives. Wearing a belt around one's waist is a risk management implementation of keeping your pants from exposing your backside. Risk management is all about covering our behinds (Hall, 1998:5). In business it becomes increasingly important to protect your interests and that of your operational units, being departments, segments or an entire company. In the end, everyone in any situation will be required to apply risk management on his immediate environment. Hall (1998:18-19) argues that, based on Deming's disciplines, risk management should manifest itself at a personal level. This is then embodied in a self-improvement process and a dedication to personal excellence.

Risk management is currently widely employed in industrial and financial circles. Many financial institutions have Financial Risk Managers who are responsible for evaluating financial transactions and investments for risk. They have to identify risks that can lead to the loss of money. On the industrial front, mining and manufacturing companies employ Occupational Risk Managers to evaluate people, processes and their environment to limit the loss of life and injury, as well as damage to property and equipment.

Risk management is a process of constant evaluation and correction. When an evaluation is performed on a situation or process, hazardous elements are to be removed or corrected to eliminate or reduce the event(s) from happening. Two types of risk management approaches exist. Proactive risk management's goal is the preemptive evaluation of processes and situations to try and identify possible risk areas to control them before or as they occur. Reactive risk management strategies have been called the "Indiana Jones school of risk management" (Pressman, 1997:133). It is based on the idea of "we'll think of something when it happens". A hazard is defined as something with the potential to cause harm, with risk the likelihood of that potential being realised. It is further put as follows: "Alternatively, hazard is the potential danger; risk is the actual danger" (Glendon & McKenna, 1995:320). That harm can come to any component inside or outside of the project or system. For open systems, the risks are so much higher.

Risk management consists of five crucial elements, being (Hall, 1998:39-40):

- Identification;
- Analysis;
- Planning;
- Tracking; and
- Resolution.

There are two primary components to every risk management process being assessment (identify, analyse) and control (plan, track, resolve).

5.3 SOFTWARE RISK MANAGEMENT

Software risk management is the practice of assessment and control on risks, which can affect a software project's success, its process or the product (Hall, 1998:8). It is therefore the application of the risk management methodology on software development, management and maintenance phases. Risk management becomes a communication tool for project managers and -members. Risk information is an indispensable resource to organise priorities and provides team members with the knowledge to make intelligent and informed decisions.

5.3.1 Risk management concepts

Hall lists the basic concepts of software risk management as:

- <u>Goal</u>
  Risk is managed relative to a specific goal which can affect only the remaining work that is required to reach the goal.

- <u>Uncertainty</u>
  Uncertainty is the unknown factor of life. We are uncertain of what we do not know. Risk occurrence always has an uncertainty connected to it. In traditional statistics, we would use probability to measure the level of (un)certainty. In risk management the likelihood of a loss to occur and the potential harm determines the relative priority of a risk.

- <u>Loss</u>
  Any form of risk will impact on loss of any resource, being time, money and people, amongst others, or the occurrence of any unfortunate situation. Without the potential of loss, there is no risk.

- <u>Time</u>
  Time cannot be accumulated, which makes it one of the more scarce resources available to software projects. Risk management helps to avoid time loss as we are prepared for unfortunate incidents. Good planning helps projects to steer around wasted time, or create alternative options to reduce the severity of time lost.

- <u>Choice</u>
  Risk management is only affective when choice is involved. By choosing alternative options during planning and management of software projects, low-risk alternatives can be evaluated to steer around high-risk situations. This is

also an important aspect of resource allocation.

The primary goal of risk management has three aspects. Firstly, it is used to make intelligent decisions. Intelligent decisions are based on information, awareness, insight and the understanding of risk. It provides a method of communicating risk information and creates insight into software risks at all project levels. Secondly, risk resolution is performed through the application of a risk action plan. In this process it is important to identify risks in time to react to it. Different risk resolution strategies may exist, ranging from minimizing risk to maximizing opportunity. Acceptable risk is defined by the decision maker. Lastly, risk management is utilised to prevent problems. Better defined, it is about controlling and limiting loss. This loss can be of any nature: Time, money or people. It is a proactive strategy to reduce the problem of costly reworks (Hall, 1998:10).

5.3.2 The value of software risk management

If the developer has time to choose to avoid high-risk situations while doing project management within a software engineering framework, there is enough time for software risk management. Software risk management is focussed on the identification of high risk areas, eliminating it in its entirety or controlling the influence it may have on the process. The value of software risk management within the software engineering framework is mainly found when dealing with uncertainty. External influences can severely affect and damage the engineering process. In the current information technology environment, it provides a mechanism to ensure the best return on investment (ROI) for both the client and the team. It limits exposure to uncertainty and is used to communicate critical information to the project team, and the client. Software risk management is based on the integrated spiral model of Barry Boehm that is iterative and risk-driven (Figure 5.5).

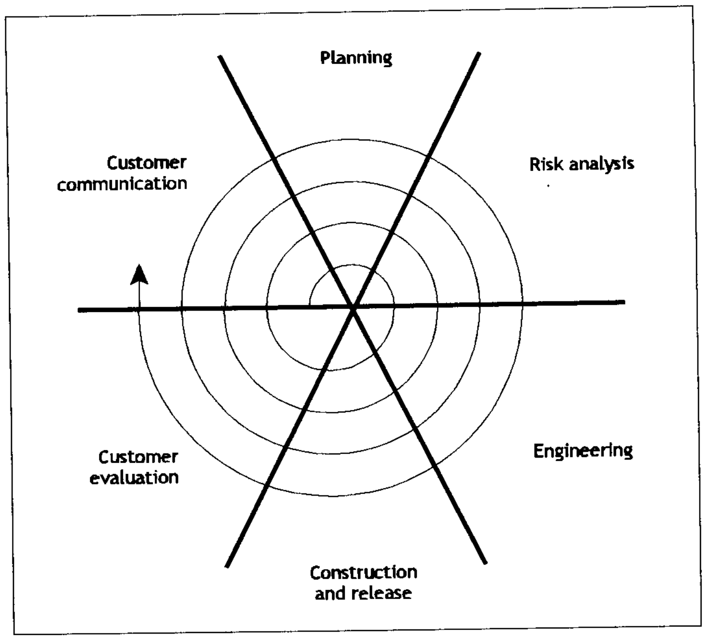


Figure 5.5 Boehm's Spiral Model

Pressman (1997:41)

5.3.3 The Software Capability Maturity Model (SW-CMM)

The Software Engineering Institute (SEI) developed the Capability Maturity Model (CMM) as a reference guide to companies to measure their software processes as it progresses towards a culture of software excellence. It provides a template that can be placed over a company's software processes through questionnaires and a five-point grading system. Companies use it primarily to measure the maturity of its processes. It is also required by some large corporations and organisations to approve vendors and service providers, e.g. the US Air Force (Hall, 1998:14).

According to Baumert & McWhinney (1992:10) this is done by looking at the issues most critical to software quality and process improvement. The CMM defines five maturity levels of software processes.

5.3.3.1 Initial

Software development is mostly ad hoc and gets chaotic at times. No formal process definitions exist and success is highly reliant on individual efforts and brilliance. There is usually a single driving force behind projects, e.g. management.

5.3.3.2 Repeatable

Basic project management principles are present keeping track of mainly cost, schedules and functionality. There are procedures in place to plan and track progress made within the project. Software configuration management is used to obtain software baselines and control the software requirements. It provides a stable and managed working environment.

5.3.3.3 Defined

Standard processes exist in an organisation for the development and maintenance of software throughout the organisation. Software engineering and management is an integrated function and a software engineering process group (SEPG) facilitates software process definition and improvement efforts. Training is facilitated through the entire organisation to produce knowledgeable and skilled staff and managers. Standardisation is also present in the use of technologies and peer review procedures are in place to improve quality.

5.3.3.4 Managed

Quantitative goals form the core of companies at this level. Productivity and quality are measured for critical software process areas of all projects around the organisation. All software processes have been configured to provide quantitative measurements that can be fed back to the control functions of the company. This is used to measure performance and quality of the company.

5.3.3.5 Optimizing

The primary focus for companies at this level is continuous improvement of development procedures. Weak process elements are identified and then strengthened and improved. The goal is to prevent the occurrence of defects. Innovations that can hold future value for the company and its processes are identified and evaluated for possible implementation.

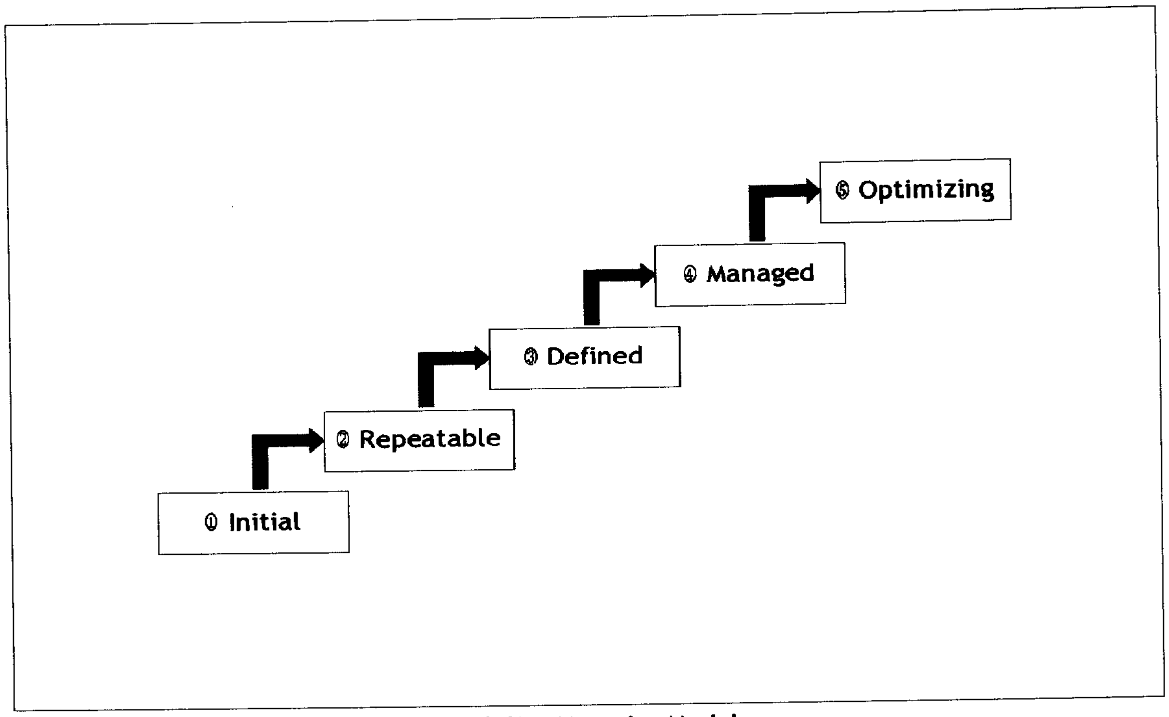


Figure 5.6 The levels of the Capability Maturity Model

The characteristics of each of these levels reiterate the control factor required in software development. The ultimate goal of this process is to provide high quality software in a controlled and measurable environment.

5.3.4 The six-discipline project management model

Extending Deming's manufacturing process model, Hall (1998:13-15) adds two further modules to the PDCA (Plan-Do-Check-Act) model. These two modules are essential to software development as it focusses on the principles of vision and discovery. This model retains the essence of the Deming model, based on quality management principles. Each of the disciplines in Hall's risk management model relates to a level in the CMM. As both of these models are built around and towards the improvement of quality, it embodies a close synergy between these two approaches. Deming's chain reaction found that higher quality can deliver higher productivity (Hall, 1998:13) which in turn can lead to higher output and better return on investment.

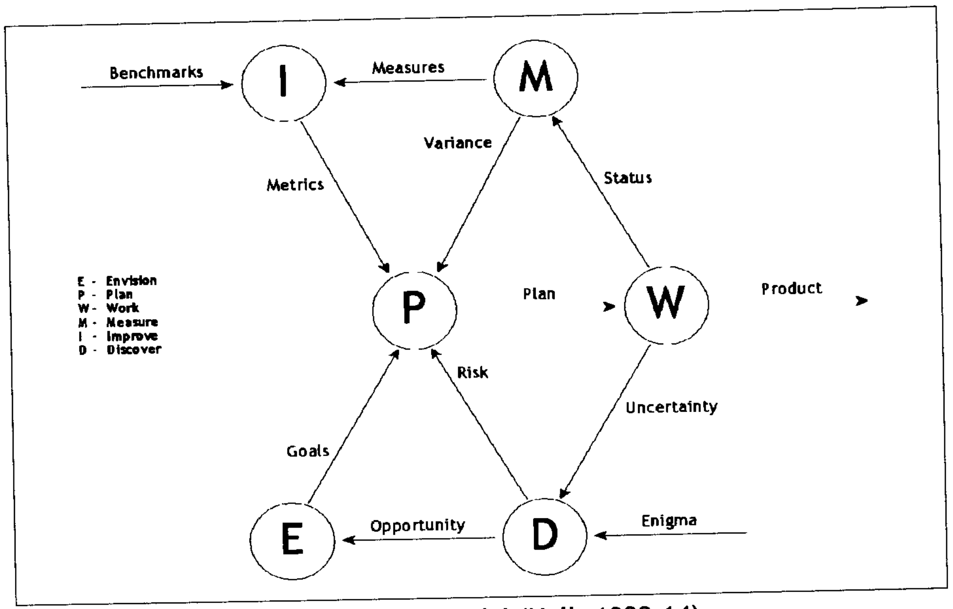


Figure 5.7 The six-discipline model (Hall, 1998:14)

5.3.4.1 Envision

Based on Deming's *chain reaction*, the six-discipline model also begins where ideas are transformed into goals and objectives. Management has to demonstrate its commitment to the vision statement, which is in turn communicated to the rest of the organisation and accepted as the new philosophy. It is always characterised by the commitment of higher powers to create. The envision-discipline delivers formal goals as output. The CMM Level 1 provides key process areas for the envision-discipline.

5.3.4.2 Plan

Project goals and objectives provide the backing for the assignment of resources to requirements. Even though overall project management is centred at the senior management, each resource[1] within the group does his own planning and management for work and tasks assigned to him. This is based on experience and knowledge obtained from previous projects. The main output of this discipline is the work plan. The CMM Level 2 provides key process areas for the Plan discipline.

1 The term resource refers to actual people as project resources

5.3.4.3 <u>Work</u>

The work-stage focusses on the implementation of the solutions formulated for the goals and requirements stated in the first principle and structured in the second discipline. It is the main production phase of the process. Three products are produced from this discipline being the product, uncertainty and status. Products remain in the work-phase until it conforms to quality requirements. Key processes for the Work discipline are found in the CMM Level 3.

5.3.4.4 <u>Measure</u>

Actual results are compared to expected results. This is used to measure performance to the expected output plan and allows for mid-process corrections. The plan is then to be updated to reflect the changes made. The CMM Level 4 provides key processes for the Measure discipline.

5.3.4.5 <u>Improve</u>

By learning from past experiences, we learn how to improve ourselves and our processes for future work. The benchmarks and project measurements are analysed for this and the information recovered from this fed back into the system. The CMM's Level 5 focusses on continual improvement. It forms therefore the best resource for implementing this discipline.

5.3.4.6 <u>Discover</u>

This discipline is not covered in the CMM, but provides a good tool for continuous improvement as well as technological and business advancement. Chapters 3 and 4 focussed on the evaluation of other unfamiliar and unknown technologies and methodologies for project improvement. We assess uncertainty of our work and environment for risk which is managed through alterations to the plan. It investigates the unknown.

5.3.5 Knowledge and ignorance in software engineering

No-one can place a price on the value of information. Even more so for knowledge, as knowledge is obtained from information. In terms of the software engineering field, knowledge also brings with it great responsibility. An alignment is required, a new state of mind. The best thing any person can do with this knowledge is to turn it into opportunities. Hall (1998:22) states that knowledge of the six-discipline model for effective behaviour requires an orientation towards applying this model as well as recognising individual intelligence.

Software projects need to incorporate the six disciplines in a structured way throughout the entire organisation (or team). Each of these people has the responsibility to exercise all six disciplines in their everyday activities. Software engineers are ultimately accountable for what they produce. Many of the global functions, i.e. management, are assigned to the appropriate specialists. Envisioning, planning, working, measuring, improving and discovery is the responsibility of each team member. Having said that, one should always take cognisance of individual strengths and weaknesses. The individual is just as important to the success of a project as the entire group. By assigning the right person to the required work improves both the individual's contribution and the total value of the team.

In order to partake in the sixth principle personal priorities have to be orientated toward utilizing creative process and risk management. Neethling (1994:11) says that there is no clear, complete definition for the term creativity. It is something that has to be discovered. To become and act creative, the individual, organisation or group has to have the will to stimulate it. It is something that can be learnt. Hall continues to state that an orientation towards repeatable creativity is required. Creativity enables us to find solutions to problems that does not seem to be so straightforward after all. The second important part is risk management. We have to be aware of the future and develop the abilities and opportunities to bring a desired future into reality. Software risks exist independently of our individual abilities, how we manage the risk affects our level of intelligence.

Ignorance, on the other hand, is an even greater threat than identified and managed risks. Knowing of risk management and not applying it is worse than not even knowing it exists. By ignoring the value of risk management, skills are lost and valuable opportunities missed. The reoccurrence of past mistakes can have devastating effects on any project as you do not apply knowledge obtained. Eventually the pain of regret will outweigh the pain of learning (Hall, 1998:23).

5.3.6 The risk management process

The risk management process consists of five key elements. Hall lists the elements as (1998:39-

40):

5.3.6.1 Identify

Identify the risks associated with the current processes and resources and find the sources of these risks.

5.3.6.2 Analyse

Use predefined criteria in analysing the risk. The probability and severity (or consequence) of each identified risk have to be measured according to the criteria set and then prioritised from the highest to the lowest. This will touch the notion of acceptable risk. Carter et al (1988:83) has a diagrammatical presentation of evaluating this.

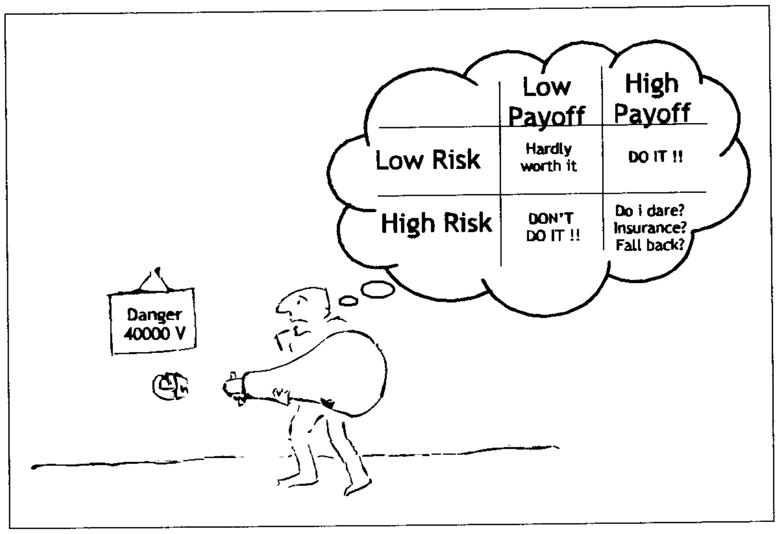


Figure 5.8 Deciding on acceptable risk
Carter et al (1988:83)

5.3.6.3 Plan

Plan alternative strategies or approaches to resolve the risks. Should you decide to accept the identified risk, create an action plan to counter and limit the influence of the risk. Establish thresholds to use as basis for the action plan's execution.

5.3.6.4 Track

Constantly monitor the risk action plan thresholds and risk status to know when to engage the action plan. Use the variance between the threshold and status as guide and implement triggers and alarms as early warning signals. Report the risk measures and metrics.

5.3.6.5 Resolve

Respond to triggers and alarms, execute the action plan at the threshold and report results and risk resolution efforts until the level of risk is acceptable.

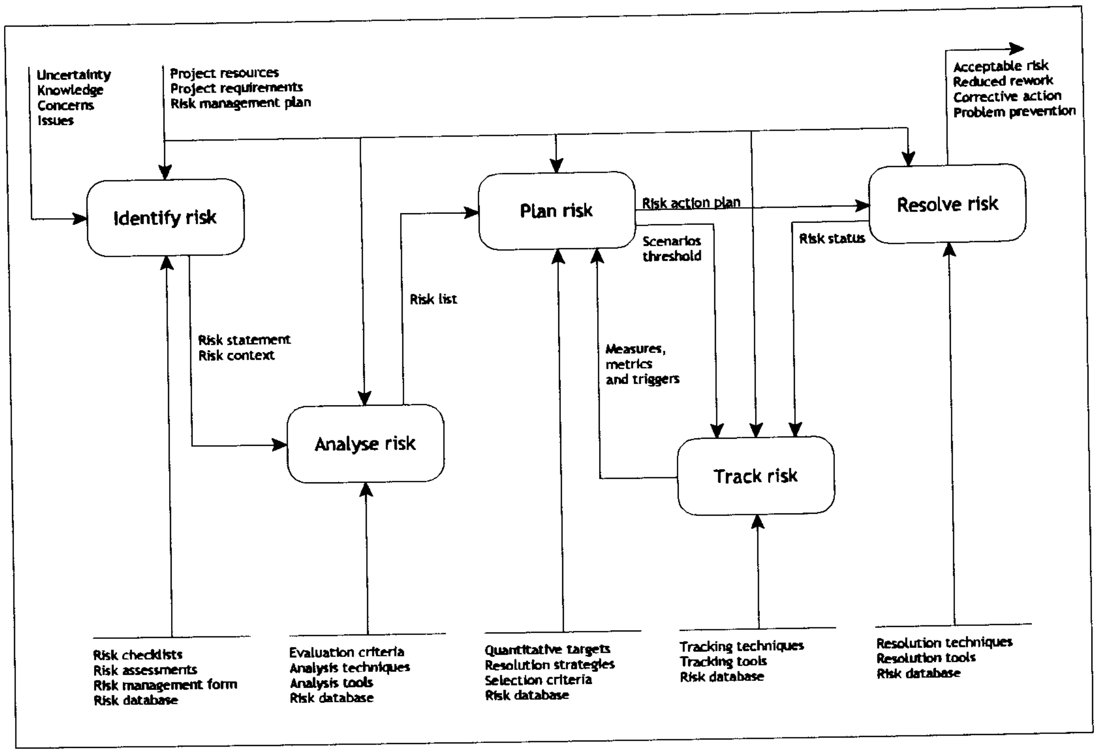


Figure 5.9 Risk management process
Hall (1998:68)

5.4 INCORPORATION OF RISK MANAGEMENT PRINCIPLES INTO SOFTWARE ENGINEERING

As stated earlier, an orientation towards risk management is required for development projects to be effective. In essence, it has to become part of one's life. Risk management is not only something that is practised at work or while driving in your car. Risk management is a concept that has to become part of everything you do. It is therefore important to incorporate risk management into current and existing development models and life cycles.

As many of the models we use today were developed when software engineering was still young and risk management was non-existent, it is important to remodel current development models to include risk management in every phase and section thereof. Risk management is not a component or module of a development life cycle. It has to become the essence of it. Figure 5.4 depicts the open system that is influenced by any incoming stimuli. It does not have any real control over what enters or exits it. It is virtually transparent and in-observable. Risk management is to provide the tool to perform certain important functions for this open system:

- It must control any external stimuli entering the system which may affect its components, resources and operation;
- It must control any outputs and information that leave the system;
- It must control the interaction of resources and procedures within the system, acting as an internal control mechanism;
- It must be the most important communication tool available to the team and its resources;
- It will be a main source of information to external agents;
- It will define the boundaries of the system, to the internal and external agents and the environment;
- It will become the primary tool to manage risks and resources; and
- It has to be transparent to all internal elements of the system, and has to be used and accessed by all elements of the system.

Figure 5.10 depicts the Risk Management Governed Open System. Risk management should be present throughout the system. Unlike the partially open system in Figure 5.3 that has a gatekeeper that controls inputs into and outputs from the system, risk management does far more than that. It also controls internal interaction and activities. It manages resources and processes to minimise risk, and therefore damage.

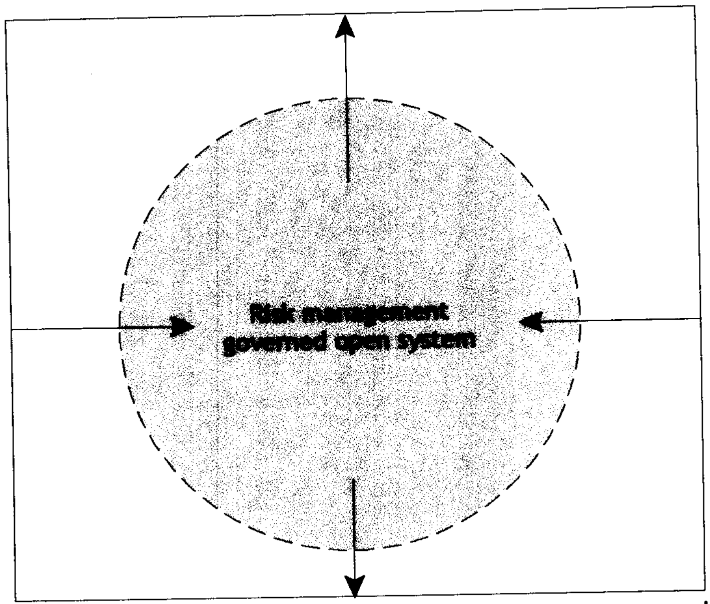


Figure 5.10 The proposed risk management governed open system

5.5 RECOMMENDATIONS FOR FUTURE STUDIES ON THE SUBJECT

The value of risk management as a management and control tool has been clearly shown in this chapter. Many procedures exist to control loss within software projects. They have been incorporated into the models they support. In order to advance software engineering principles to the next level of maturity, providing high quality software and better return on investment for clients and business, risk management has to be incorporated into existing development models. For software risk management to achieve its full potential, it has to be taken apart and analysed in depth. The methodology has to be adapted to fit into almost any system and structure used to develop software, without being a quick-fix plugin onto the side of development models, but as an extension to all internal processes and procedures. The requirements listed in paragraph 5.4 will provide the backbone for the design of models and the application of risk management principles.

Unfortunately the in-depth investigation and development of risk management principles fall outside the scope of this study. This study serves as a platform for future studies into the combined subjects of software engineering and software risk management principles.

5.6 CLOSING REMARKS

This study showed the important impact of software evaluation as a tool for decision making within software engineering. Even though it is widely seen as a luxury, or a waste of time, software engineers have to realise the importance it can have in the development of high quality software. Performance measurement should however not be seen as the most important gauge for project success.

Developing quality software is a way of life. Quality has to be a passion and the force that drive software engineering. Quality and productivity are not competing for business's top requirements in software systems. Quality can lead to higher productivity. Risk management has proven itself in finance and industrial areas. It has to become the governing force within software projects, teams and organisations.

BIBLIOGRAPHY

Books

AHITUV, N., NEUMANN, S., RILEY, H.N. 1994, *Principles of Information Systems for Management*. Dubuque: Business and Educational Technologies.

BEHLENDORF, B., CHANDLER, D.M., BRINTLE, L., CASSELBURY, R., ANTHONY, T., DARNELL, R., ESTABROOK, N., GRABER, J., HUBBARD, C., LADD, E., PARKER, R., SCOTT, C.A. 1996, *Running a perfect web site with Apache*. Que: Indianapolis.

Borland Delphi 6 Developers Guide 2001. Scotts Valley, California: Borland Software Corporation.

BROOKSHEAR, J.G. 1994, *Computer Science: An overview*. Redwood City: The Benjamin/Cummings Publishing Company.

CARTER, R., MARTIN, J., MAYBLIN, B., MUNDAY, M. 1994. *Systems, Management and Change: A Graphic Guide*. London: Harper & Row.

FENTON, N.E., PFLEEGER, S.L. 1997, *Software Metrics: A Rigorous and Practical Approach*. Second edition. Cambridge: Cambridge University Press.

GLENDON, A.I., MCKENNA, E.F. 1995, *Human safety and Risk Management*. London: Chapman and Hall.

HALL, E.M. 1997, *Managing Risk: Methods for software systems development*. Upper Saddle River: Addison-Wesley.

HAMBURG, M., YOUNG, P. 1994, *Statistical Analysis for decision making*. Sixth edition. Fort Worth: The Dryden Press.

HUYSAMEN, G.K. 1983, *Inferensiële Statistiek en Navorsingsontwerp: 'n Inleiding*. Derde hersiene uitgawe. Pretoria: Academia.

JACOBSON, I., ERICSSON, M., JACOBSON, A. 1995, *The Object Advantage: Business Process Re-engineering with object technology*. Wokingham: Addison-Wesley.

KAST, F.E., ROSENZWEIG, J.E. 1985, *Organization and Management: A Systems and Contingency Approach*. Fourth edition. New York: McGraw-Hill.

MARX, S., RADEMEYER, W.F., REYNDERS, H.J.J 1992. *Bedryfsekonomie: Riglyne vir Ondernemingsbestuur*. Pretoria: J.L. van Schaik.

MINK, O.G., SHULTZ, J.M., MINK, B.P. 1986, *Developing and managing open organisations*. Texas: Organization and Human Resource Development Associates.

NEETHLING, K. 1994, *Kreatiwiteit laat jou wondere verrig*. Pretoria: Benedic Boeke.

PRESSMAN, R.S. 1997, *Software Engineering: A Practitioner's Approach*. Fourth edition. New York: McGraw-Hill.

SPSS Base 7.0 for Windows User's Guide 1996. Chicago: SPSS Inc.

STALLINGS, W. 1995. *Operating systems*. Second edition. New Jersey: Prentice-Hall.

STEYN, A.G.W., SMIT, C.F., DU TOIT, S.H.C., STRASHEIM, C. 1994, *Moderne Statistiek vir die praktyk*. Pretoria: J.L. van Schaik Uitgewers.

WHITTEN, J.L., BENTLEY, L.D., BARLOW, V.M. 1994. *Systems Analysis and Design Methods*. Third edition. Boston: Richard D. Owen.

WILLCOCKS, L., SAUER, C. 2000, *Moving to e-business*. London: Random House.

Bibliography

Articles

BAUMERT, J.H., MCWHINNEY, M.S. 1992, *Software Measures and the Capability Maturity Model.* Software Engineering Institute. Carnegie Mellon University.

BOTHA, W.M., DU TOIT, P.H. 1999, *Guidelines for the preparation of written assignments".* University of Pretoria.

Notes to a technical presentation the August 1998 Inprise Conference. *Ten thinks you can do to make InterBase scream.* Presented by Karwin, B., 1998.

RITCHLEY, R.W., HAMILTON, B.A. 2001, *Open Source vs. Close Source Software: An experiment to determine which is more secure.*

Software Engineering Institute. 1995. *Perspectives on Legacy System Reengineering.* Carnegie Mellon University.

Journals

GELSINGER, P.P., GARGINI, P.A., PARKER, G.H., YU A.Y.C 1989. Microprocessors: circa 2002. *IEEE Spectrum*, October 1989, p. 43-47.

MOORE, G.E. 1965. Cramming more components onto integrated circuits. *Electronics,* April 1965, vol. 38, no. 8.

OGLETHORPE, S. 2002. The elements of an Enterprise Portal. *Sun Journal,* May 2002, vol. 5, no 3.

OUSTERHOUT, J.K. 1998. Scripting: Higker level programming for the 21st century. *IEEE Computer*, March 1998.

World Wide web sites

ABREU, E.M. 1999. "Microsoft: Bad Security, or bad press?", http://europe.cnn.com/TECH/computing/9909/28/ms.security.idg/index.html (November 2002).

BALTAZAR, H., DYCK, T. 2001. "Tux: Built for speed", http://cma.zdnet.com/texis/techinfobase/techinfobase/+bwq_qr+ss6+6+/zdisplay.html (November 2002).

BOZDOC, M., "The main events in computing history", http://www.bozdoc.f2s.com/ (January 2001).

Borland 2002. "Delphi Awards", http://www.borland.com/delphi/awards/index.html (November 2002).

Borland 2002. "Kylix Awards", http://www.borland.com/kylix/awards/index.html (November 2002).

BURNS, J. 2002. "Java vs. JavaScript", http://htmlgoodies.earthweb.com/beyond/j_vs_js.html (November 2002).

DRAGOI, O.A. "The Conceptual Architecture of the Apache Web Server", http://www.math.uwaterloo.ca/~oadragoi/CS746G/a1/apache_conceptual_arch.html (November 2002).

Economic Research Institute 2002. "Kolmogorov-Smirnov Test", http://www.eridlc.com/onlinetextbook/appendix/table7.htm (November 2002).

GLASNER, J., "Tech Meltdown: Lessons Learned?", http://www.wired.com/news/business/0,1367,42169,00.html (November 2002).

GAUDET, D. 2001, "Apache Performance Notes", http://httpd.apache.org/docs/misc/perf-tuning.html (November 2002).

HONTAÑÓN, R.J. 2000. "Emerging Technology: Maximizing Apache Server Security", http://www.networkmagazine.com/article/NMG20000710S0016 (November 2002).

LEINER, B.M., CERF, V.G., CLARK, D.D., KAHN, R.E., KLEINROCK, L., LYNCH, D.C., POSTEL, J., ROBERTS, L.G., WOLF, S., "A Brief History of the Internet", http://www.isoc.org/internet/history/brief.shtml (November 2002).

MACVITTIE, L. 2001. "Tips for Choosing a Web Server", http://dcb.sun.com/practices/howtos/webserver_part1.jsp (November 2002).

Microsoft 2001. "Windows 2000 Server Family: Delivering the Level of Reliability You Need", http://www.microsoft.com/windows2000/server/evaluation/business/overview/reliable/default.asp (November 2002).

Microsoft 2002. "Internet Information Server Resource Kit: Chapter 4 - Performance Tuning and Optimization", http://www.microsoft.com/technet/prodtechnol/iis/reskit/iis40rg/iisrkc04.asp (November 2002).

Netcraft Web Server Survey 2002. http://www.netcraft.com/survey (November 2002).

PIENAAR, I. 1999. "Outrage at Microsoft's independent, yet sponsored NT 4.0/Linux research", http://www.itweb.co.za/sections/enterprise/1999/9904221410.asp (November 2002).

Perl Mongers 2002. "Perl Fast Facts", http://www.perl.org/press/fast_facts.html (November 2002).

RAYMOND, E.S. 1999. "The Mindcraft fiasco", http://features.slashdot.org/article.pl?sid=99/04/23/1316228&mode=thread&tid=109 (November 2002).

Thames Valley University 2002. "Guidelines on reference listing - The Harvard system", http://www.tvu.ac.uk/lrs/guides/harvard.html (November 2002).

The Apache Software Foundation 2002. "Apache HTTP Server Project", http://httpd.apache.org/ABOUT_APACHE.html (November 2002).

ULICK, J., "Wall St.: From bad to worse", http://money.cnn.com/2001/12/25/markets/markets_review/index.htm (November 2002).

Appendix 1 The Critical values of T for the Wilcoxon matched-pairs rank test

Level of significance for one-tailed test					Level of significance of one-tailed test				
	0.05	0.025	0.01	0.005		0.05	0.025	0.01	0.005
Level of significance for two-tailed test					Level of significance for two-tailed test				
N	0.10	0.05	0.02	0.01	*N*	0.10	0.05	0.02	0.01
5	0	-	-	-	28	130	116	101	91
6	2	0	-	-	29	140	126	110	100
7	3	2	0	-	30	151	137	120	109
8	5	3	1	0	31	163	147	130	118
9	8	5	3	1	32	175	159	140	128
10	10	8	5	3	33	187	170	151	138
11	13	10	7	5	34	200	182	162	148
12	17	13	9	7	35	213	195	173	159
13	21	17	12	9	36	227	208	185	171
14	25	21	15	12	37	241	221	198	182
15	30	25	19	15	38	256	235	211	194
16	35	29	23	19	39	271	249	224	207
17	41	34	27	23	40	286	264	238	220
18	47	40	32	27	41	302	279	252	233
19	53	46	37	32	42	319	294	266	247
20	60	52	43	37	43	336	310	281	261
21	67	58	49	42	44	353	327	296	276
22	75	65	55	48	45	371	343	312	291
23	83	73	62	54	46	389	361	328	307
24	91	81	69	61	47	407	378	345	322
25	100	89	76	68	48	426	396	362	339
26	110	98	84	75	49	446	415	379	355
27	119	107	92	83	50	466	434	397	373

(Hamburg & Young, 1994:A-24)

Appendix 2 Kolmogorov-Smirnov Test

Sample size	Level of significance				
N	.20	.15	.10	.05	.01
1	.900	.925	.950	.975	.995
2	.684	.726	.776	.842	.929
3	.565	.597	.642	.708	.828
4	.494	.525	.564	.624	.733
5	.446	.474	.510	.565	.669
6	.410	.436	.470	.521	.618
7	.381	.405	.438	.486	.577
8	.358	.381	.411	.457	.543
9	.339	.360	.388	.432	.514
10	.322	.342	.368	.410	.490
11	.307	.326	.352	.391	.468
12	.295	.313	.338	.375	.450
13	.284	.302	.325	.361	.433
14	.274	.292	.314	.349	.418
15	.266	.283	.304	.338	.404
16	.258	.274	.295	.328	.392
17	.250	.266	.286	.318	.381
18	.244	.259	.278	.309	.371
19	.237	.252	.272	.301	.363
20	.231	.246	.264	.294	.356
25	.210	.220	.240	.270	.320
30	.190	.200	.220	.240	.290
35	.180	.190	.210	.230	.270
Over 35	$1.07/\sqrt{N}$	$1.14/\sqrt{N}$	$1.22/\sqrt{N}$	$1.36/\sqrt{N}$	$1.63/\sqrt{N}$

Economic Research Institute

(http://www.eridlc.com/onlinetextbook/appendix/table7.htm)

Appendix 3 Percentiles in the upper regions of the standard-normal distribution

Cumulative probability	z	Cumulative probability	z	Cumulative probability	z
0.50	0.0000	0.70	0.5244	0.90	1.2816
0.51	0.0251	0.71	0.5534	0.91	1.3408
0.52	0.0502	0.72	0.5828	0.92	1.4051
0.53	0.0753	0.73	0.6128	0.93	1.4758
0.54	0.1004	0.74	0.6433	0.94	1.5548
0.55	0.1257	0.75	0.6745	0.95	1.6449
0.56	0.1510	0.76	0.7063	0.96	1.7507
0.57	0.1764	0.77	0.7388	0.97	1.8808
0.58	0.2019	0.78	0.7722	0.975	1.960
0.59	0.2275	0.79	0.8064	0.98	2.0537
0.60	0.2533	0.80	0.8416	0.99	2.3263
0.61	0.2793	0.81	0.8779	0.995	2.5758
0.62	0.3055	0.82	0.9154	0.999	3.0902
0.63	0.3319	0.83	0.9542		
0.64	0.3585	0.84	0.7745		
0.65	0.3853	0.85	1.0364		
0.66	0.4125	0.86	1.0803		
0.67	0.4399	0.87	1.1264		
0.68	0.4677	0.88	1.1750		
0.69	0.4959	0.89	1.2265		

(Huysamen, 1983:211)

Appendix 4 Glossary of acronyms

API	Application Program(ming) Interface
ARPANET	Advanced Research Projects Agency Network
ASP	Active Server Pages
CGI	Common Gateway Interface
CLF	Common Logfile Format
CMM	Capability Maturity Model
CSS	Cascading Style Sheet
DLL	Dynamic Link Library
DSO	Dynamic Shared Object
DTP	Desktop publishing
FORTRAN	Formula Translation/Translator (high level language)
FTP	File Transfer Protocol
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IDE	Integrated Development Environment

IEEE	Institute of Electrical & Electronics Engineers
IIS	Internet Information Server
IP	Internet Protocol
ISAPI	Internet Server Application Programming Interface
IT	Information Technology
LAN	Local Area Network
NCSA	National Center for Supercomputing Applications
NIC	Network Interface Card
ODBC	Open Database Connectivity
PERL	Practical Extraction and Reporting Language
PWS	Personal Web Server
RAD	Rapid Application Development
ROI	Return on Investment
RPM	Redhat Package Manager
SDLC	Systems Development Life Cycle
SEI	Software Engineering Institute (Carnegie Mellon)
SEPG	Software engineering process group

SQL	Structured Query Language
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TPS	Transactions per second
UNIVAC	Universal Automatic Calculator
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
XML	eXtensible Markup Language