ABSTRACT

CHARACTERIZING THE EVOLUTION OF OPERATING SYSTEMS

By
Yi Peng

Examining the development and trends in software engineering technology is a huge undertaking. It is constantly evolving and affected by a large number of factors, which are themselves driven by a wide range of sub-factors. This dissertation is part of a long term project intended to analyze software engineering technology trends and how they evolve. This project is intended to analyze operating system trends and what are the factors that drive how they evolve. Basically, the following questions will be answered: "How to watch, predict, adapt to, and affect operating system's evolution trends?"

In previous research, YF Chen used statistical models to analyze the evolution of programming languages. Building upon Chen's work, the author uses operating systems as the subject, derives the statistical models and applies them to analyze the trend and the relationships between different factors that characterize an operating system.

After the history of several operating systems is reviewed, it shows that two kinds of factors, intrinsic factors and extrinsic factors, could affect the evolution of an operating system. Intrinsic factors are used to describe the general design criteria of an operating system. On the other hand, extrinsic factors are the factors that are not directly related to the general attributes of an operating system. In order to describe the relationship of these factors and how they affect operating system trends, they need to be quantified. For intrinsic factors, data are collected from different trustable data sources and analyzed. For extrinsic factors, historical data are collected and established as a data warehouse. The

operating system trends are described and evaluated by using all the data that have been collected and analyzed.

In this dissertation, statistical methods are used to describe historical operating system trends and predict the future trends. Several statistics models are constructed to describe the relationships among these factors. Canonical correlation is used to do the factor analysis. Multivariate multiple regression method has been used to construct the statistics models for the evolution of operating system trends. The models are validated by comparing the predicted data with the actual data.

CHARACTERIZING THE EVOLUTION OF OPERATING SYSTEMS

by
Yi Peng

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Science

Department of Computer Science

January 2005

APPROVAL PAGE

CHARACTERIZING THE EVOLUTION OF OPERATING SYSTEMS

Yi Peng

Dr. Ali Mili Dissertation, Dissertation Advisor (date)
Professor of Computer and Information Science, NJIT

Dr. Joseph Leung, Committee Member (date)
Distinguished Professor of Computer Science, NJIT

Dr. Dimitri Theodoratos, Committee Member (date)
Associate Professor of Computer Science, NJIT

Dr. Vincent Oria, Committee Member (date)
Assistant Professor of Computer Science, NJIT

Dr. Fu Li, Committee Member (date)
PhD of Mathematics, NJIT

BIOGRAPHICAL SKETCH

Author: Yi Peng

Degree: Doctor of Philosophy

Date: January 2005

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2004

- Master in Computer Engineering,
  Northeastern University, Shenyang, P.R. China, 1999

- Bachelor in Computer Engineering,
  Northeastern University, Shenyang, P.R. China, 1996

Major: Computer Science

Publications and Presentations:

Yi Peng, Fu Li, Ali Mili, Shaheen Zainuddin. "Modeling the Evolution of Operating Systems: An Empirical Study", *Submit to IEEE Transactions on Software Engineering 2004.*

Yi Peng, Michael Halper, Yehoshua Perl, and James. Geller. "Auditing the UMLS for redundant classifications", *Proceedings of the 2002 American Medical Informatics Association (AMIA) Annual Symposium.*

Hellen. Gu, Yehoshua Perl, G. Elhanan, Hua. Min, Li. Zhang, Yi Peng. "Auditing Concept Categorizations in the UMLS". *Artificial Intelligence in Medicine. (31) 1, pages 29-44, May 2004.*

Yehoshua. Perl, Zong. Chen, Michael. Halper, James. Geller, Li. Zhang, and Yi. Peng. "The cohesive metaschema: A higher-level abstraction of the UMLS Semantic Network". *Journal of Biomedical Informatics, 35(3), pages 194-212, June 2003.*

Hellen Gu, Hua Min, Yi Peng, Li. Zhang, and Yehoshua Perl. "Using the metaschema to audit UMLS classification errors". *In Proceedings of the 2002 American Medical Informatics Association (AMIA) Annual Symposium, pages 310-314, San Antonio, TX, November 2002.*

Yi Peng, Xiaohu Zhang, Chuanxiang Rong, Guirang Chang, "On Query Improvement in Web Search Engine", *China's CERNET'98 Conference Proceedings.*

Presentation: Software Architecture, *NJIT 2004.*

Presentation: Software Trend Watch, *NJIT 2004.*

Presentation: Auditing the UMLS for redundant classifications. *San Antonio, TX, November 2002.*

Presentation: Using the metaschema to audit UMLS classification errors. *San Antonio, TX, November 2002.*

Participation: SNOMED Users Group Conference. *San Antonio, TX, November 2002.*

To my parents

for letting me pursue my dream

for so long and so far away from home

To my husband Chen Zhang

For all his support through all these years

To myself

For all persistency

ACKNOWLEDGMENT

First and foremost, I would like to thank my advisor, Professor Ali Mili, for his support over the years. His advice on technical matters was invaluable, and his guidance on the Operating system evolution project was critical to the project's success.

I would like to thank the members of my thesis committee: Prof. Ali Mili, Prof. Joseph Leung, Prof. Dimitri Theodoratos, Prof. Vincent Oria and Dr. Fu Li, for their valuable feedback and advice.

I would like to thank Shaheen for his wonderful and very diligent efforts in developing the web pages for this project.

Also, early in the years of achieving my Master's degree, Dr. Guiran Chang gave tremendous help and invaluable instructions. His patience and advice always encourages me.

I would like to thank my family, for being very supportive and patiently listening to me trying to explain my research. I would like to thank my parents for believing me every time I said I was graduating "next year."

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF TABLES
(Continued)

LIST OF FIGURES

CHAPTER 1

SOFTWARE ENGINEERING TRENDS

1.1 Introduction

Software engineering is an engineering discipline whose goal is the cost-effective development of software systems. It was first proposed in 1968 at a conference held to discuss what was then called the "software crisis". Tremendous progress has been achieved since 1968 and the development of the software engineering discipline has markedly improved the practice of software development. A much better understanding of the activities involved in software development has been observed in past years. Although there is a lot of progress, many researchers still consider software engineering as a relatively young discipline. After a period of research about software engineering, there is more and more interest in the evolution of software engineering.

Predicting the evolution of software engineering technology is, at best, a dubious proposition. The recent evolution of software technology is a prime example; it is fast paced and affected by many factors, which are themselves driven by a wide range of sources. Many of the factors are outside the arena of software engineering and most of them cannot even be identified. Right now, this doctoral work is at the early, and tentative, stage of a project to analyze software engineering technology trends and how they evolve. In this dissertation, the author will discuss the tentative venture in this domain and sketch prospects for future research.

The purpose of this project is to analyze technology trends and try to gain some insight into how they evolve. While this doctoral dissertation is at the very early, and

very tentative, stages of the whole project, research approaches of this problem could be characterized by two premises:

1. Structuring the problem. When approaching the problem of software engineering technology watch, there are many questions that need to be known. All of these questions are interrelated: some questions refine others; some questions complement others; some questions provide the background for others; some questions overlap with others, etc. The first order of business, for this project, is to build a questionnaire structure, which arranges all these questions in a way that attempts to highlight their interrelations. Also, questionnaire structure should be improved by refining questions that are too vague, merging identical questions, or synthesizing related questions.

2. Specifying the solution. Three research methods can be deployed: analytical research, which attempts to understand the phenomena that underlie observed behavior, and build models that capture these phenomena: empirical research, which makes no attempt to understand cause/effect relationships, but merely attempts to capture observed behaviors by empirical models; experimental research, which intervenes after analytical or empirical research to validate the proposed models. For each issue, it is useful to deploy a judicious combination of these three methods.

1.2 Questionnaire Structure

To focus the effort on specific issues and to lend some structure to this inherently complex problem, a questionnaire has been built on a hierarchy of the following questions:

1. How to watch software engineering trends? This question deals with what indicators are needed to monitor, where to find them, and how to interpret them.

2. How to predict software engineering trends? This question deals with what life cycle do software engineering trends follow, and what triggers the passage of a trend from one phase to another along the lifecycle.

3. How to adapt to software engineering trends? This question deals with how does one define institutional strategy in such a way as to maximize benefit from what is known about a trend and minimize risk from what is not known about it.

4. How to affect software engineering trends? Perhaps more crucial is whether trends can in fact be affected by any single entity. This question tries to identify where, in the cycle of a trend, is it possible to alter the course of the trend, and eventually how and by whom.

At the center of this hierarchy is the question of how to predict software engineering trends. If this question can be understood well, the others can be answered with adequate precision.

1.3 Watching Software Engineering Trends

The general goal of watching software engineering trends is to determine what information must be maintained in order to gain a comprehensive view of the discipline and its evolution. The information in question must be sufficiently rich to support discipline-wide assessments as well as trend-specific analysis. The following questions will be asked to watch software engineering trends:

- What is the relevant information that must be collected/monitored?
- Where this information could be found, or where did it infer from?
- How to interpret this information?
- How often does it need to update this information?

A number of software engineering-specific and technology-generic indicators have been identified, which have been divided into the following categories: classification Standing, Research and Development, Science and Technology Output, Human Resources, Costs and Funding, Standards and regulations, and Best Practices.

1.4 Predicting Software Engineering Trends

The general goal of predicting software engineering trends is probably the most important and the most difficult goal of this whole study. The focus of this goal is on identifying a lifecycle that trends follow. Once this lifecycle is identified, the software engineering trends can be predicated.

- Research Trends, which are a favorite topic of panel sessions and surveys.
- Technology Trends, which are driven by the maturation of applicable research ideas, and by the successful evolution of the idea to a useful, technologically viable, product.
- Market Trends, which are created either by the supply side or by the demand side in different situations.

For different trends, different methods will be used to analyze and predict them. In this point of view, empirical methods should be used to analyze research trends and technology trends, analytical research should be used for market trends. There are already some good analytical models for market trends, such as the Chasm Theory by Geoffrey A. Moore. This tentative research concentrates on research trends and technology trends.

1.5 Adapting to Software Engineering Trends

The general goal of adapting to software engineering trends is: "how to adapt to a trend if the trend has been known?" For example, a corporate manager hears about a particular trend (e.g. XML, .NET, Linux) and wants to know what to do about it: Ignore it? Adapt the corporate products and support it? Develop a new set of products that support it? Etc. When a party has particular stake in the evolution of a trend, he/she may need a distinct information profile to make a judicious decision.

Adapting to technology trends depends to a large extend on watching technology trends and on predicting technology trends. If a corporate manager wants to make a decision on a given trend, what does he/she need to know about it? It is recognized that several features must be analyzed and/or quantified in order to provide support for this kind of decisions:

- What are the stakes of this trend for the organization?
- What are the intrinsic technical merits of this trend?
- What are the adoption costs of this trend?
- What are the adoption risks of this trend?
- What are the adoption benefits of this trend?
- How long is the trend expected to have an impact?
- What is the optimal time to make an adoption decision?

1.6 Affecting Software Engineering Trends

In this aspect of the project, there is an interest in analyzing to what extend it is possible to affect/control technology trends. It is not sufficient to have an impact on a trend. It is more necessary that the impact can be premeditated and preplanned.

Detailing the general goals discussed above, the following questions have been derived:

- Is it possible to affect technology trends?
- Who can affect technology trends?
- How can technology trends be affected?
- How to quantify the impact?

1.7 Conclusion

Software engineering trends are briefly discussed in this chapter. The following is what need to be done in this tentative effort.

Because the problem is formulated in all its generality, a bottom up approach is adopted. Similar as what has been done in YF Chen [1], in this project, the following work is done:

- First, try to formulate the problem of technology watch in terms of a hierarchy of increasingly specific questions. This hierarchy of questions serves two purposes: first to focus the effort on specific issues that need to be addressed; second to lend some structure to this inherently complex problem.

- Second, show how a systematic combination of empirical, experimental, and analytical approaches can give the researchers means to gain some understanding of the problem. Analytical approaches, mostly inspired from earlier work, will be used to derive candidate models for the complex evolution of software engineering trends; empirical approaches will be used to derive evolutionary models, or model aspects, without emphasis on analytical explanation of the models; and experimental approaches will be used to collect the necessary data to fill in the parameters of the candidate models and to test them for adequacy.

CHAPTER 2

FOCUS ON A FAMILY OF TRENDS: OPERATING SYSTEMS

2.1 Introduction

As the early stage of the software engineering trends project, it is a good practice to concentrate on one particular field first. For example, what is the evolution process of programming languages, computer networking, or databases? Some good practice in programming languages have been done before. Now this dissertation will focus on a family of software engineering trends: operating system trends.

Why should the operating system be selected as an example for software engineering trends project? The followings are the reasons:

- First, there is no modern software if there is no operating system. It is well known that operating system is the most fundamental system program which controls all the computer's resources and provides the base upon which the application programs can be written.

- Second, without the development of operating system, there is not development of software engineering. The development of operating system took a very important role in the development of software engineering. By reviewing the history of operating systems, a somewhat clearer view on software engineering could emerge.

- Third, hundreds of operating systems were created in the past, some of them were very successful, and some of them failed although a lot of resources were spent on them.

- Fourth, they represent a unity of purpose and general characteristics, across several decades of evolution.

- Fifth, they offer a wide diversity of features and a long historical context, thereby enabling precise analysis.

- Sixth, their history is relatively well documented, and their important characteristics relatively well understood.

For the above reasons, it is clear that there is a good set and a rich history of operating systems that are well worth discussing. Therefore, operating systems will be used as a good sample set for this research project.

2.2 History of Operating Systems

Operating system is an extended machine. It is the most fundamental system program which controls all the computer's resources and provides the base upon which the application programs can be written. Operating systems have been evolving through the years. The development of operating systems has been historically closely tied to the architecture of the compute on which they run [2].

The first generation computers (1945~55) are vacuum tubes and plug boards. Operating systems were unheard of, even assembly languages were unknown.

The second generation computers (1955~65) are transistors and the corresponding solution operating systems are batch systems. The idea behind it was to collect a tray full of jobs in the input and then read them onto a magnetic tape using a small, relatively inexpensive computer, such as IBM 1401. The ancestor of today's operating system is to read and run one of the jobs. Large second-generation computers were used mostly for scientific and engineering calculations, such as solving partial differential equations. They were largely programmed in FORTRAN and assembly language. Typical operating systems were FMS (the FORTRAN Monitor System) and IBSYS, IBM's operating system for the 7094 [2,3].

The third generation operating system (1965~1980) are ICs and multiprogramming operating systems. One of the leading one is IBM OS/360 (also known as System/360). It

was designed for a series of software-compatible machines from 1401-sized to much more powerful than the 7094. Furthermore, the OS/360 was designed to handle both scientific and commercial computing. The 360 line was the first major computer line to use Integrated Circuits (ICs), thus providing a major price/performance advantage over the second generation computers [2,4].

The fourth-generation computers (1980~Present) are personal computers. With the development of LSI (Large Scale Integration) circuits, chips containing thousands of thousands transistors on a square centimeter of silicon, the age of the personal computer dawned. The most powerful personal computers are usually called workstations. Two operating systems initially dominated the personal computer and workstation scene: Microsoft's MS-DOS and UNIX. Later on, Microsoft also has a series of operating systems from Windows 95, Windows 98, Windows NT, and Windows 2000 to Windows XP, the latest version of Windows. And UNIX derives UNIX based operating systems like Solaris, FreeBSD, OpenBSD, Netware, etc. and the newest member of UNIX family --- Linux [2,5].

An interesting development that began taking place during the mid-1980s is the growth of networks of personal computers running network operating systems and distributed operating systems. In a network operating system, the users are aware of the existence of multiple computers and can log in to remote machines and copy files from one machine to another. Each machine runs its own local operating system and has its own local user.

A distributed operating system, in contrast, is one that appears to its users a traditional uni-processor system, even though it is actually composed of multiple

processors. The users should not be aware of where their programs are being run or where their files are located; that should all be handled automatically and efficiently by the operating systems.

2.3 Operating Systems Trends

Because operating system trends are part of software engineering trends, similar methods will be used to analyze them. This dissertation will concentrate on how to watch, predict, adapt to, and affect operating system trends.

The methods of how to watch, predict, adapt to, and affect software engineering trends have been discussed in Chapter 1. The historical trends of operating system will be discussed in the similar way. After having better understanding on the past trends, the author will concentrate on how to predict operating system trends. Empirical method will be used in this project.

To research the operating system trends, the following questions will be discussed:

- Is it possible to predict if an operating system will succeed or fail?
- What will be the operating systems that will be studied?
- What are the possible factors which can affect the trend?
- What information should be collected in order to determine if an operating system succeed or fail?
- How to quantify the factors and find a model/function to predict the trends?
- What are the results of the evolution analysis for operating system?
- Beyond the analysis of the evolution of individual operating system, this project also analyzes the evolution of the features. So that, even if whether an operating system is successful or not could not be predicted, the main attributes that affect the future of operating systems can be characterized.

In this dissertation, all of these questions will be discussed. The answers of all these questions will form an outline for the whole operating system trends.

2.4 Research Methods

In this dissertation, the author will concentrate on how to watch, predict, adapt to, and affect operating system trends. From the evolution of software technology, the author thinks this evolution is affected by a dizzying array of factors, which are themselves driven by a wide range of sources. Monitoring operating system trends is not as untraceable as it may sound, that it does not have to be an ad-hoc, erratic process.

The following is the process that are adopted in this project:

- Find out the possible factors that maybe affect operating system trends
- Quantify those factors
- Analyze the history of operating system
- Build statistical model(s) to watch evolution of operating system
- Predict the future trends of operating system
- Validate the statistical model(s)

Although it is impossible to find out "exact accurate" model for operating system trends, these model are useful and can be used to describe the history of operating system. By extending the historical models, it could also be used to predict the future evolution of operating system.

CHAPTER 3

SELECTING RELEVANT FACTORS

What are the possible factors that can affect the trend of operating systems? This is the question which will be discussed in Chapter 3. To answer this question, research should be done in both the internal properties of operating system themselves and the outside world which may have some influences on the operating system trends. Although it is not sure how they could affect the operating system.

In previous research, the author has identified that two kinds of factors, intrinsic factors and extrinsic factors, could affect the operating system trends. Section 3.1 will discuss what intrinsic factors are and how to identify those factors. Section 3.2 will discuss extrinsic factors via the same methods.

When choosing the factors, the following criteria are followed: completeness, orthogonality and general significance. For completeness, the author would like to make sure that the chosen factors are complete to describe an operating system. The purpose is to find out a useful framework which can be used to evaluate an operating system. Orthogonality means that each factor is concentrating on one particular part of an operating system. Different factors are dealing with different aspects. General significance insures that the factors are fair for every operating system and no factors are "designed" for a specific operating system.

Two quantitative factors are summarized to characterize an operating system:

- Intrinsic factors that describe the technical features of an operating system and are usually time independent.

- Extrinsic factors that describe the outer environment in which the OS exists and evolves and are typically time relevant.

3.1 Intrinsic Factors

The following 19 factors have been identified as intrinsic factors of operating system. These factors will be divided into categories: Resource Management; Usability; Usefulness from Functional point of view; Usefulness from Operational point of view; Versatility; Design; Cost. These factors will be discussed in the following.

3.1.1 Resource Management

1. Scalability

Scalability is an operating system's ability to increase its processing capacity as CPUs are added. If the processing capacity increases in direct proportion to the number of CPUs, a system is said to exhibit 100% scaling. In practice, a system's ability to scale is limited by contention between the CPUs for resources and depends on the mix of applications being run [6]. Scalability encompasses a broad range of issues, but the development community has primarily targeted these areas for improvements:

- Providing better administrative tools for managing very large installations
- Improving the scalability characteristics of the operating system and removing the architectural constraints from the kernel
- Optimizing the system throughput to accommodate enterprise-class networking
- Expanding support for high-end hardware solutions[7]

This is a very significant factor of operating systems as modern multi-processor computing getting more and more popular.

2. CPU Management

CPU management is an operating system's ability to mange the CPU resource. CPU resource management is commonly known as scheduling. The scheduling policy is determined by the way the computer will be used, although most policies can use a common scheduling mechanism. This mechanism determines how CPU will be allocated to processes and the policy determines the order in which ready processes will receive services. A group of CPU management methodologies are considered such as First-Come-First-Served, Shortest Job Next, Priority Scheduling, Deadline Scheduling, Round robin, Multiple-level Queues, Multiple-Processor Scheduling, Real-Time Scheduling [3,4,5,8]. CPU is one of the most important resources and the scheduling policy determines the way the computer will be used.

3. Memory Management

Memory is an important resource that must be carefully managed. The memory hierarchy is composed with several parts: a small amount of very fast, expensive, volatile ache memory; medium-speed, medium-price, volatile main memory (RAM); and slow, cheap, nonvolatile disk storage. It is the job of the operating system to coordinate how these memories are used. The part of the operating system that manages the memory hierarchy is called the memory manager. Its job is to keep track of which parts of memory are in use and which parts are not in use, to allocate memory to processes when they need it and de-allocate it when they are done, and to manage swapping between main memory and disk when main memory is too small to hold all the processes. There are kinds of memory management methods that have been applied by a lot of operating systems. [2,3,4,5,8,9,10].

4. I/O Management

One of the main functions of an operating system is to control all the computer's I/O (Input/Output) devices. It must issue commands to the devices, catch interrupts, and handle errors. It should also provide an interface between the devices and the rest of the system that is simple and easy to use. To extend the possible, the interface should be the same for all devices (device independence). The I/O code represents a significant fraction of the total operating system [4]. I/O management plays a key role in that it provides an interface between the devices and the rest of the system that is simple and easy to use.

3.1.2 Usability

Usability property of operating system shows the degree of how easy an operator can operate the system. There are three different aspects for usability.

1. Ease of Learning

Ease of learning involves effort required to master the interfaces provided by the system in terms of formal schooling, on the job training and associated misuse of the system at various intervals in the learning curve [2].

2. Ease of Use

Ease of use is the property to indicate the ease of operating the operating system. Christian Green said: "It takes more than a graphical user interface to make a computer easy to use. It takes tight integration between software and hardware. It requires an operating system that's graphical 'from the ground up', so that users don't have to deal with character-based code. And it requires a company that focuses on the user, and helps guide software developers to make the user experience more consistent." [11]

3. Consistency of Interaction Protocols

Consistency of interaction protocols refer to the conformity of descriptions of standard patterns of interaction between a human being and a computer system [12]. It is of great magnitude for an operating system to provide consistent interaction protocols for user.

3.1.3 Usefulness from Functional Point of View

The property of usefulness indicates the how the operating system is useful for the operators. There are two parts for usefulness: functional and operational.

1. Range of Services

Operating system services are responsible for the management of platform resources, including the processor, memory, files, and input and output. They generally shield applications from the implementation details of the machine. There are a lot of services provided by operating systems, like kernel operations, command interpreter,utility services, batch processing services, File and directory synchronization services [13].

2. Range of Programming Languages Support

After reviewing the brief history and language features of each programming language, the author decide to investigate the operating system's support within the following set of programming languages: Ada, ALGOL, Pascal, C, C++, COBOL, FORTRAN, Java, Perl, LISP.

3. Distributed Computing

Distributed computing is a programming model in which processing occurs in many different places (or nodes) around a network or across a facility. Processing can

occur wherever it makes the most sense, whether that is on a server, web site, personal computer, handheld device, or other smart device [14].

4. Network Services

Network services are provided to support distributed applications requiring data access and applications interoperability in heterogeneous or homogeneous networked environments. A network service consists of both an interface and an underlying protocol [13]. In the era of network, the network services are very important for an operating system.

5. Deadlock Management

Deadlock management is the ability for operating systems to detect, prevent, avoid and recover from deadlock. Deadlock is a significant problem that can arise in a community of cooperating or competing processes. Therefore, deadlock management feasibility is an important factor of operating systems

3.1.4 Usefulness from Operational Point of View

1. Reliability

Reliability is the ability of a system perform its required functions under stated conditions for a specified period of time [6,15]. Reliability is generally considered important by end users. Not all companies making operating systems have a similar standard. Even among operating systems where reliability is a priority, there is a range of quality. Also, an operating system may be extremely reliable at one kind of task and extremely unreliable at another. For example, reliability includes following features: Stability, Back-ups, Error reporting, Fail over, Hot-swapping hardware.

2. Security & Protection

Security and Protection is the ability of a system to manage, protect, and distribute sensitive information. There are kinds of methods to implement security and protection features [13]. Throughout the discussion of the operating system managers, various ways to protect resources from unauthorized access have been mentioned --- protection and security are pervasive in the operating system [3].

3.1.5 Versatility

1. Portability

Portability is the ease with which an operating system can be transferred from one hardware or software environment to another [6]. For example, diverse hardware support and File system support methodologies are aspects for portability properties. At a time when different computer lines of the same vendor didn't talk to each other --- but alone machines of multiple vendors --- portability means a great savings in both hardware and software upgrades and thus are very important operating system factor.

2. Compatibility

Compatibility is the ability of two or more operating systems to perform their required functions while sharing the same hardware or software environment [6]. This could include Upward Compatibility and Downward Compatibility. On computers that contain multiple operating systems, compatibility becomes more complex and important.

3. Openness

Openness is the degree to which an operating system complies with standards [6]. For openness property of operating system, following features should be considered:

- Open source

- Open system:
    - the use of interface standards [16]
    - the use of implementations that conform to those standard interfaces[16]
- Network management [16]
- Protocols

It establishes no sharp boundary between the OS itself and the user's programs, and the techniques used to make the system robust. Therefore it is an important feature of operating system.

3.1.6 Design

Design of integrity indicates the ability an operating system omit certain anomalous features and improvements but to reflect one set of design ideas, than to have on that contain many good but independent and uncoordinated ideas [17]. The design principles combine traditional wisdom with extensions to address the evolution of future interfaces. Existing design principles are based on IBM's experiences in user interface design, on the design experiences of others, and on insights from linguistics and psychology. These design principles have been extended to address evolving interfaces that will provide a more friendly appearance and behavior in the future. The increasing use of 3-D and real-world representations as well as the blossoming popularity of the Internet and the World Wide Web has strongly influenced these progressions [18].

There are several design principles such as:

- Simplicity: Don't compromise usability for function. Keep the interface simple and straightforward.

- Support: Place the user in control and provide proactive assistance. To give users control over the system, enable them to accomplish tasks using any sequence of steps that they would naturally use.

- Familiarity: Build on users' prior knowledge. Allow users to build on prior knowledge, especially knowledge they have gained from experience in the real world.

- Obviousness: Make objects and their controls visible and intuitive. Where you can, use real-world representations in the interface.

- Encouragement: Make actions predictable and reversible. A user's actions should cause the results the user expects.

- Satisfaction: Create a feeling of progress and achievement. Allow the user to make uninterrupted progress and enjoy a sense of accomplishment.

- Availability: Make all objects available at all times. Users should be able to use all of their objects in any sequence and at any time.

- Safety: Keep the user out of trouble. Users should be protected from making errors.

- Versatility: Support alternate interaction techniques. Allow users to choose the method of interaction that is most appropriate to their situation. Interfaces that are flexible in this way are able to accommodate a wide range of user skills, physical abilities, interactions, and usage environments.

- Personalization: Allow users to customize. The interface should be tailorable to individual users' needs and desires. No two users are exactly alike.

- Affinity: Bring objects to life through good visual design. The goal of visual design in the user interface is to surface to the user in a cohesive manner all aspects of the design principles.

3.1.7 Cost

Most technology committees concentrate their budget and planning efforts on the acquisition of next-generation software and hardware. But hardware and software costs represent a small fraction of the total expense of networked computers. Accordingly, this

focus overlooks the most important factors driving the rising costs of computer systems. The neglected costs are management and labor expense [19].

Which is cheaper? Hardware costs, software licenses, technical support agreements, prices of upgrades/service packs, costs of hardware upgrades, profits lost for every hour of downtime, personnel costs for recovering/recreating data lost due to product defects in the operating system and/or hardware platform required by your choice of operating systems, these are only some of the factors that contribute to the overall budget resulting from the decision. It is not a trivial consideration. Although money is the bottom line for a manager, given the complex set of factors The author have just presented, a technically superior combination of server hardware and operating systems could prove to be less expensive in the long run [20].

3.2 Extrinsic Factors

Intrinsic factors are the attributes of an operating system itself. After a set of intrinsic factors is determined, the following questions are raised out:

1) Are intrinsic factors enough to determine the future of an operating system?

2) Are there any other factors which can also affect the trend of an operating system?

3) If the answer is "Yes" in (2), what should be the other factors?

Naturally, the outside world will also have impact on the trend of an operating system. Next, the following question is explored: "What could be the possible factors which can affect the trend of an operating system?"

Extrinsic factors are the factors which are not directly related to the general attributes of an operating system, but still can affect the trend of operating system.

The purpose is to find out as many as possible factors from "outside world" which may affect the trend of an operating system. In order to find a desirable set of extrinsic factors, aspects in the real world are checked. Right now, the extrinsic factors are classified into the following seven categories: Institutional Support, Industrial Support, Governmental Support, Organizational Support, and Grassroots Support. Each category includes several questions.

I. Institutional Support

1. Support the OS: the institutional unit provides the environment for a given operating system and allows people using it.

2. Teach using the OS: the lecturers in the institutional unit use the operating system during their teaching process.

3. Teach the OS: the lectures in the institutional unit teach a given operating system in a course.

4. Research using the OS: in the institutional unit, a given operating system is used in the research activities.

5. Research on the OS: in the institutional unit, a given operating system is a research object.

Here, institutional unit could be colleges, universities, research center, lab, academic center and other institutional units.

II. Industrial Support

1. Not Support using OS: the industrial unit does not provide the environment for a given operating system and does not allow employee using it.

2. Support using OS: the industrial unit provides the environment for a given operating system and allows employee using it.

3. Encourage using the OS: the industrial unit encourages the usage of a given operating system within the unit.

4. Require using the OS: the industrial unit requires the usage of a given operating system within the unit.

III. Governmental Support

1. Not Support the OS: the governmental unit does not provide the environment for a given operating system and does not allow employee using it.

2. Support the OS: the governmental unit provides the environment for a given operating system and allows employee using it.

3. Encourage using the OS: the governmental unit encourages the usage of a given operating system within the unit.

4. Require using the OS: the governmental unit requires the usage of a given operating system within the unit.

IV. Organizational Support

1. Is this OS introduced and supported by any (international) organization?

2. Are there any organization standards?

3. How many conferences for this OS?

4. How many conference papers/articles are published on this OS?

5. How many conference papers/articles are published by using OS?

V. Grassroots Support

1. Know the OS: people are aware of a given operating system.

2. Use the OS: people use a given operating system.

3. Prefer to use the OS: people prefer to use a given operating system.

In the author's point of view, both intrinsic factors and extrinsic factors could impact on the evolution of operating systems. Identifying these factors is the first step for this empirical study.

CHAPTER 4

QUANTIFYING THE FACTORS

In this chapter, after selecting a set of factors, How to quantify the factors is discussed. Quantifying means assigning a numeric function to each factor.

4.1 Methods to Quantify the Factors

All of the factors should be considered when one designs an operating system. So, all features of an operating system should be checked to see if it matches these factors.

Several methods are introduced to quantify the selected factors.

4.1.1 Numeric Formula

The first group factors have been given a numeric formula which is understandable and widely acceptable. For example, the average number of system failures per month is used to quantify *Reliability*. Another example here is that the quantifying method for *Cost* is the prices in U.S. dollars for U.S. delivery. Except these two factors, *ease of learning*, *portability*, *compatibility* and *organizational support* can be categorized into the same group in regards of its quantifying method. This method is very straightforward and easy to understand.

4.1.2 Hierarchical Sub-features

The second set of factors use different quantifying methods. In order to quantify these factors, a set of discrete features that are usually associated with the factor are chosen, respectively. Then rank these features from 1 (lowest) to N (highest), where N is the

number of features into a partial hierarchical level. The score of an operating system is then derived as the sum of all the scores that correspond to the features it has.

For example, in order to quantify memory management, 11 sub-factors are taken into consideration, and range them from *garbage collection* (score:1) to *shared-memory multiprocessor* (score:6), as shown in Table 4.1. Some of the sub-features may have same score because they are in the same level in the hierarchy. For example, *both variable partition memory strategy* and *address translation* have been assigned 2, but they belong to different categories. For a particular operating system, all the sub-factors are looked into, and then the final score for this operating system is derived by summing up the scores of all the supported sub-factors.

Table 4.1 Hierarchical Sub-feature Quantifying Methods: Memory Management

Memory Management	Features	Scores
	Garbage collection	1
Memory Allocation Strategies	Fixed-partition Memory strategy	1
	Variable partition memory strategy	2
	Contemporary Allocation Strategy	3
	Runtime bound Checking	4
Memory Management Strategies	Swapping	1
	Address Translation	2
	Static Paging	3
	Dynamic Paging	4
	Segmentation	5
	Shared-memory multiprocessor	6

For detail, check the next section. The author acknowledges that this method is controversial as it may sound arbitrary; but the author finds it adequate for the current purposes, as it generally reflects the intuition about how candidate operating systems compare with respect to the intrinsic factors.

In this group, included intrinsic factors are: *scalability*, *CPU management*, *I/O management*, *range of services*, *distributed computing*, *network services*, *deadlock management*, *security and protections*, *openness* and *institutional support*.

4.1.3 Cumulative Sub-features

Similar to hierarchical sub-factors, for the third group of factors, a set of distinct features that are usually associated with this factor are also picked. But instead to assign an order to them, they are considered to be of equal quantifying. The final score will be the number of features that an OS has taking into account that one feature contributes one point and so on.

For example, ten programming languages are summarized to quantify the factor of *range of programming languages support*. ten programming languages are in the investigated set. For a given OS, if it supports one programming language, it will get 1 point. Then the final score for this OS will be the number of all the programming languages it supports. The same methodology is applied for *consistency of interaction protocols*.

4.1.4 Discrete Scale Sub-features

The fourth category of factors, again, has a list of separate features. Rather than order them into a hierarchy or just cumulate the number, for a given OS, each of the features will be qualified according to a set of mutual exclusive scales. A score is assigned to these scales from "excellent" to "not good" as 5 to 1. Therefore, the score for the feature is the score of the corresponding scale. Thus, the score for the upper factor is the average of all these features scores.

For example, eight sub-features are used to quantify *ease of use*. (For detail, see next section) For each of the eight sub-features, its score is decided by comprehending which scale it is provided. If, for a particular OS, it provides "excellent" *help and manual feature*, which is one of the eight sub-features for *ease of use*. Five points are added to the final score for this particular operating system on *ease of use*. Again, it provides "very good" *multimedia support*, another sub-feature, thus turns a four points. After decide all the feature points, the scores from all the sub-feature and summed up, and divided by the number of the sub-features (here, it's eight), thus the final score of the factor is obtained.

All the eight sub-features are listed in following. Besides *ease of use*, the same methodology to quantify *design* is applied.

4.1.5 Exclusive Rating Sub-features

This method is applied on most of the extrinsic factors. Most of the questions are asking for the numbers. But different questions have different priorities. For example, *Governmental Supports* has four different behaviors: not support, support, encourage, require. For the author's point of view, they contribute different priority points for scoring the governmental support. Encourage means that the a governmental unit encourages the use of the OS, which means that compare to another governmental unit that does not encourage (e.g., only support or even not support) the use of the OS, the first one "add" more point to the final score of Governmental support than the later one for this particular OS. In other words, for a particular OS, if more governmental units encourage using it than a second OS, it can be concluded that the first OS gets more points than the later one on the aspects of encouraging using. Similarly, require contributes more than encourage for the final score.

Based on these observations, four level supports are sorted from weakest to strongest and assign them points. Not support gets 0, support gets 1, encourage gets 2, require gets 3. Note that the four supports are defined in an exclusive fashion. For example, if one OS gets 2 for governmental support, which means that a governmental unit encourages to use the OS, in other words, the unit already supports to use the OS, then it will get 2 points rather than 1 point, because 2 is enough to explain the support level. In this way, for a particular OS, the average score of all governmental support is used as the final score.

Same methodology described above is applied on *Industrial Support*, *Governmental Support* and *Grassroots Support*.

In Table 4.2, all the factors and their corresponding quantifying methods are listed.

4.2 Quantifying the Factors

In this section, the author will check each factor and discuss how to quantify it.

1. Scalability: All the sub-factors are considered and sorted from weakest to the strongest one, as shown in Table 4.3 [7].

2. CPU Management: All the sub-factors are considered and sorted from weakest to the strongest one, as shown in Table 4.4 [2,3,4,5,8,9,21].

3. Memory Management: All the sub-factors are considered and sorted from weakest to the strongest one, as shown in Table 4.5 [3,4,14,22,23,24].

4. I/O Management: All the sub-factors are considered and sorted from weakest to the strongest one, as shown in Table 4.6 [3,4,5].

Table 4.2 Quantifying Methods

Factor	Quantifying Methods				
	Numeric Formula	Hierarchical Sub-features	Cumulative Sub-features	Discrete Scale Sub-features	Exclusive Rating Sub-features
Scalability		√			
CPU Management		√			
Memory Management		√			
I/O Management		√			
Ease of Learning	√				
Ease of Use				√	
Consistency of Interaction Protocols			√		
Range of Services		√			
Range of Programming Languages Support			√		
Distributed Computing		√			
Network services		√			
Deadlock Management		√			
Reliability	√				
Security & Protection		√			
Portability	√				
Compatibility	√				
Openness		√			
Design				√	
Cost	√				
Institutional Support		√			
Industrial Support					√
Governmental Support					√
Organizational Support	√				
Grassroots Support					√

Table 4.3 Features used to quantify Scalability

Scalability	Score
Input/output Handing	1
Interrupt Binding	2
Administrative Tools	3
Process Binding	4
Kernel Asynchronous I/O	5
Direct I/O	6
Scalable TCP/IP Networking	7
Multithreaded IP Stack Operations and Asynchronous I/O	8

Table 4.4 Features used to quantify CPU Management

CPU Management	Score
First-Come-First-Served	1
Shortest Job Next	2
Priority Scheduling	3
Deadline Scheduling	4
Round Robin	5
Multiple-level Queues	6
Multiple-Processor Scheduling	7
Real-Time Scheduling	8

Table 4.5 Features used to quantify Memory Management

Memory Management	Features	Scores
	Garbage collection	1
Memory Allocation Strategies	Fixed-partition memory strategy	1
	Variable partition memory strategy	2
	Contemporary allocation strategy	3
	Runtime bound checking	4
Memory Management Strategies	Swapping	1
	Address translation	2
	Static paging	3
	Dynamic paging	4
	Segmentation	5
	Shared-memory multiprocessor	6

5. Ease of Learning: The number of average hours to learn this system, including in terms of formal schooling, on the job training and associated misuse of the system at various intervals in the learning curve, are used. Obviously, the lower the number is, the better the ease of learning property for the operating system.

Table 4.6 Features used to quantify I/O Management

I/O Management	Score
Direct I/O with Polling	1
Direct Memory Access	2
Interrupt-driven I/O	3
Memory-mapped I/O	4
Kernel I/O Subsystem	5
STREAMS	6

Table 4.7 Features used to quantify Ease of Use

Ease of use Level	Score
excellent	5
very good	4
good	3
acceptable	2
not good	1

6. Ease of Use: The level for the property of ease of use. Some criteria are listed in order to judge the level of ease of use [25].

- Text based command shells vs. Graphic command shells
- Multimedia
- Animation
- The look and feel of the interface
- Simple internet access
- Quick setup
- Simple operation
- Help and manual

According to the ability of complying with these criteria, five levels of ease of use are categorized from strongest to weakest: excellent, very good, good, acceptable and not good. And assign them scores from 5 to 1 as shown in Table 4.7.

7. Consistency of Interaction Protocols: Total numbers of interaction protocols that are consistent the bigger the better. A list of interaction protocols are summarized and are applied to all the operating systems that are chosen [26].

(1) Formulate queries/Interpret responses: Uniform across all queries/responses.

(2) Command format are identical for all commands.

(3) File names: All the files in an OS have the same format.

(4) Interface consistency: The "look and feel" of the interface is consistent. For example, there is a common background and/or color scheme for all screens; common screen layouts.

(5) Function/Appearance consistency: All object appear the same, function the same. For example, "Help", "Exit", and "Search" objects are in the same spot on all screens. They have the same design, color, etc.

(6) Text characteristics consistency: The text characteristics are constant from screen to screen. For example, Ariel 14 point bold italics are always used only for chapter titles. Blue underlined text always represents a hyperlink.

(7) Semantic characteristics consistency: Metaphors and icons are used consistently throughout the interface. For example, a magnifying glass means the same thing every place it is used; a green cat always means Help.

(8) Navigation consistency: Navigation objects and steps are consistent throughout the interface. Screens are linked consistently.

(9) Interaction tools consistency: Interaction tools like mouse pointers, touch screens, joysticks are used consistently.

(10) Conventions consistency: Conventions are familiar to the user employed consistently.

(11) Screen configurations consistency: All related items on the screen are grouped together visually in a format that makes sense.

(12) Labels consistency: Labels on buttons, menus, and titles are used consistently.

(13) The number of interaction protocols that a particular operating system has implemented are added up and assigned as the grade for this operating system.

8. Range of Services: Number of Services supported. Table 4.8 lists the basic services that an operating system should provide. In order to quantify the services property for every operating system, the corresponding score is assigned for every service according to their features [13].

Table 4.8 Features used to quantify Range of Services

System Services	Score
Create and manage processes and threads of execution	1
Execute programs	1
Define and communicate asynchronous events	1
Define and process system clock operations	1
Implement security features	1
Manage files and directories	1
Control input/output processing to and from peripheral devices	1
Object request broker (ORB) services	2
Common object services	2
Comparing, printing, and displaying file contents	3
Editing files	3
Searching patterns	3
Evaluating expressions	3
Logging messages	3
Moving files between directories	3
Sorting data	3
Executing command scripts	3
Local print spooling	3
Scheduling signal execution processes, and	3
Accessing environment information.	3
Batch processing services	4
File and directory synchronization services	4

9. Range of Programming Languages: The number of programming languages supported is used as the method to quantify the characteristics of range of programming languages supported. The number of programming languages that a particular operating system supports are added up and used to evaluate the range of PL factor. The set of programming languages used is: Ada, Algol, C, C++, PASCAL, COBOL, FORTRAN, JAVA, Perl, LISP.

10. Distributed Computing: Number of Distributed Computing Features Supported. Table 4.9 lists the basic distributed computing features that an operating system can provide. In order to quantify the property for every operating system, the corresponding score is assigned for every feature according to their characteristics [13].

Table 4.9 Features used to quantify Distributed Computing

Distributed Computing	Score
Distributed time	1
Distributed data	2
Distributed file	3
Distributed name	4
Location	5
Remote process	6
Remote print spooling and output distribution	7

11. Network services: The cumulated number of network services supported is used as the method to quantify the feature of network services supported by an operating system. Table 4.10 lists the basic network services that an operating system should provide. In order to quantify the networking property for every operating system, the corresponding score is assigned for every network service according to their features from strongest to weakest [27].

Table 4.10 Features used to quantify Network Services

Network Service	Score
Connectivity: Network Interface Cards, Cabling, Hubs, Bridges, etc.	1
Basic Network Services: File service, Print service, Naming service, Security, etc.	2
Network Management & Administration: Diagnostics, Monitoring, Testing, Metering, Inventory, etc.	3
Office Automation: Databases, Spreadsheets, Word Processors, Personal Management.	4
Communications: Dial-up connections, Mainframe access, FAX servers, etc.	5
Group Applications: Email, Groupware, etc.	6
Mission Critical Applications: Company Specific Applications	7

12. Deadlock/Starvation Management: Deadlock management levels [4].

Table 4.11 Features used to quantify Deadlock/Starvation Management

Deadlock Level	Score
Deadlock detection	1
Deadlock recovery	2
Deadlock prevention	2
Deadlock avoidance	3

13. Reliability: Average number of system failures during a specific time zone.

14. Security & Protection: Security and protection services supported level [4].

Table 4.12 Features used to quantify Security & Protection

Security & Protection	Score
Identification and authentication	1
System entry control services	2
Audit	3
Access control	4
Non-repudiation	5
Security management	6
Trusted recovery	7
Encryption	8
Trusted communication	9

15. Portability: Average number of man-month to transfer the system from one hardware or software environment to another.

16. Compatibility: Average number of man-month to upgrade or downgrade an operating system.

17. Openness: Cumulated levels of public standards that the system complies. A list of 23 standards that could be implemented is considered. Here is the list:

A. Single Unix Specification

B. POSIX 1 Library functions i.e. kernel calls

C. POSIX 2 Shell and utilities

D. Pthreads IEEE POSIX 1003.1c.

E. XNFS X/Open Network File System

F. X Window System Protocol

G. Xlib - C Language X Interface

H. X Toolkit Intrinsic - C Language Interface

I. Inter-Client Communication Conventions Manual

J. Motif 1.2 IEEE Std 1295

K. CDE Common Desktop Environment

L. OSI network

M. Netware Protocol

N. SNA

O. TCP/IP

P. Ipv4

Q. Ipv6

R. TCP

S. UDP

T. ICMP

U. DLPI

V. NetBIOS

W. RPC

According to the relationships between each other, the hierarchy of these standards is constructed, as shown in Figure 4.1. It can be seen that standard of "Pthreads IEEE POSIX 1003.1c." is a subset of standard "POSIX 1 Library functions i.e. kernel calls". And furthermore, "POSIX 1 Library functions i.e. kernel calls" is a subset of standard "Single Unix Specification". There are similar situation for most of other standards in the

list. Standard V --- NetBIOS and standard E --- XNFS X/Open Network File System are independent standards that are not subset and neither superset of any other standards.

By analyzing the standards hierarchy, different scores are assigned for different standards according to their position in the hierarchy, as shown in Table 4.13. For example, Pthreads IEEE POSIX 1003.1c. (D) has a point of 1. This means that there is no standard that is a subset of D. And D is a comparative simple and more specific standard that cover a small range. Standard OSI network (L) is assigned the highest points because

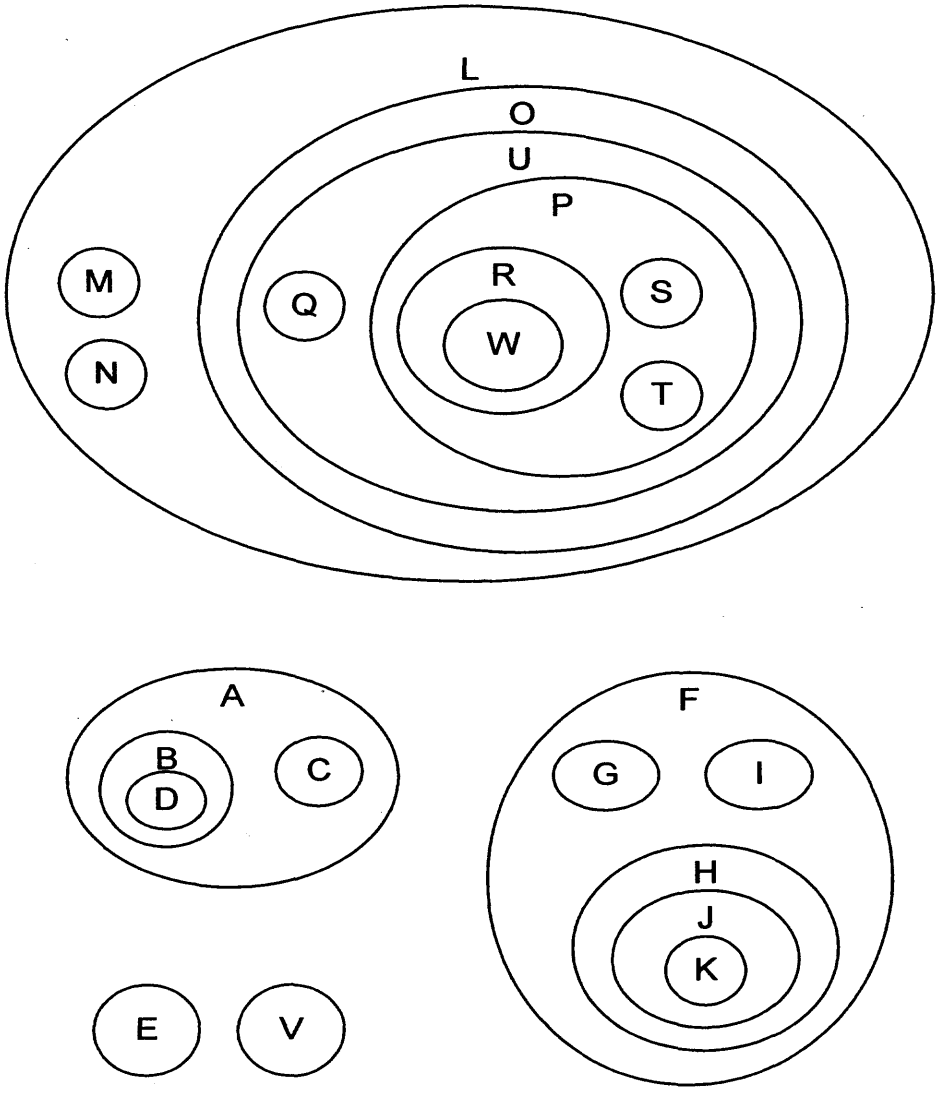


Figure 4.1 Standards Hierarchy for Openness.

it is the super set of O (TCP/IP) which has point of 5. The broader scope that the standard has covered, the higher the points that has assigned to it. For each operating system, according to its implementation for every standard, a score is given. For example, UNIX has implemented every standard and it gets a score of 47 [20,28,29,30,31,32,33,34,35].

Table 4.13 Features used to quantify Openness

Openness Standards	Score
Single Unix Specification	3
POSIX 1 Library functions i.e. kernel calls	2
POSIX 2 Shell and utilities	1
Pthreads IEEE POSIX 1003.1c.	1
XNFS X/Open Network File System	1
X Window System Protocol	4
Xlib - C Language X Interface	1
X Toolkit Intrinsic - C Language Interface	3
Inter-Client Communication Conventions Manual	1
Motif 1.2 IEEE Std 1295	2
CDE Common Desktop Environment	1
OSI network	6
Netware Protocol	1
SNA	1
TCP/IP	5
Ipv4	3
Ipv6	1
TCP	2
UDP	1
ICMP	1
DLPI	4
NetBIOS	1
RPC	1

18. Design: According to the degree of how a given operating system complies with the design principles, it is categorized to one of the five levels, from "excellent" to "not good" [18].

19. Cost: There are three issues to consider in cost: Acquisition, Maintenance and Operation.

1) Acquisition. This part indicates the cost required to obtain an operating system or set up an operating system. It includes three different types of costs:

- hardware costs: prices in U.S. dollars for U.S. delivery
- software costs: prices in U.S. dollars for U.S. delivery
- end use costs: prices in U.S. dollars for U.S. delivery

Table 4.14 Features used to quantify Design Principles: Survey Results

Design Principles	excellent	very good	good	acceptable	not good
Simplicity	5	4	3	2	1
Support	5	4	3	2	1
Familiarity	5	4	3	2	1
Obviousness	5	4	3	2	1
Encouragement	5	4	3	2	1
Satisfaction	5	4	3	2	1
Availability	5	4	3	2	1
Safety	5	4	3	2	1
Versatility	5	4	3	2	1
Personalization	5	4	3	2	1
Affinity	5	4	3	2	1

3) Maintenance. Maintenance cost is the overall cost of maintaining a computer system to include the costs associated with personnel, training, maintenance control, hardware and software maintenance, and requirements growth. Four types of costs compose the maintenance cost:

- User support costs: profits lost for every hour of downtime, hiring and paying personnel to support
- Requirement growth: prices of upgrades/service packs
- Hardware maintenance: costs of hardware upgrades
- Software maintenance: technical support agreements

3) Operation. Operation cost is the overall cost of operating a computer system to include the costs associated with personnel, training, and system operations. It has three types:

- Communication costs: prices in terms of man-month
- Development costs: prices in terms of man-month
- Downtime costs: prices in terms of man-month

In the next chapter, the chosen operating systems are discussed.

CHAPTER 5

WATCHING OPERATING SYSTEMS

To watch and predict the trends of operating system, a set of operating system should be selected as sample. By analyzing this set of operating system, statistics models will be constructed to describe the past trends of operating system. By extending the statistics models, they will also be sued to predict the future trend of an operating system.

In this chapter, the following fifteen of operating system will be investigated: UNIX, Solaris/Sun OS, BSDs (including FreeBSD, OpenBSD and NetBSD), OS/2, Windows, MS-DOS, MAC OS, Linux, NetWare, HP-UX, GNU Hurd, IBM AIX, Compaq/DEC VMS, Multics, OS360.

5.1 Unix

UNIX history goes back to 1969 and the famous "little-used PDP-7 in a corner" on which Ken Thompson, Dennis Ritchie and others started work on what was to become UNIX. The name "Unix" was intended as a pun on Multics (and was written "Unics" at first -- UNiplexed Information and Computing System) [29].

For the first 10 years, UNIX development was essentially confined to Bell Labs. These initial versions were labeled "Version n" or "Nth Edition" (of the manuals), and were for DEC's PDP-11 (16 bits) and later VAXen (32 bits). Some significant versions include:

- V1 (1971): 1st UNIX version, in assembler on a PDP-11/20. Included file system, fork(), roff, ed. Was used as a text processing tool for preparation of patents. Pipe() appeared first in V2.

- V4 (1973): Rewritten in C, which is probably the most significant event in this OS's history: it means UNIX can be ported to a new hardware in months, and changes are easy. The C language was originally designed for the UNIX operating system, and hence there is a strong synergy between C and UNIX.

- V6 (1975): First version of UNIX widely available outside Bell Labs (esp. in universities). This was also the start of UNIX diversity and popularity. 1.xBSD (PDP-11) was derived from this version. J. Lions published "A commentary on the Unix Operating System" based on V6.

- V7 (1979): For many, this is the "last true Unix", an "improvement over all preceding and following Unices". It included full K&R C, uucp, Bourne shell. V7 was ported to the VAX as 32V. The V7 kernel was a mere 40 Kbytes.

These Vn versions were developed by the Computer Research Group (CRG) of Bell Labs. Another group, the Unix System Group (USG), was responsible for support. A third group at Bell Labs was also involved in UNIX development, the Programmer's WorkBench (PWB), for example, sccs, named pipes and other important ideas. Both groups were merged into Unix System Development Lab in 1983 [36].

Work on UNIX continued at Bell Labs in the 1980s. The V series was further developed by the CRG (Stroustrup mentions V10 in the 2nd edition of his book on C++). The company now responsible for Unix (System V) is called Unix System Laboratories (USL) and is majority-owned by AT&T. Novell has bought USL (early 93)! Novell has given rights to the "UNIX" trademark to X/Open (late 93).

5.2 Solaris/Sun OS

Sun's implementation of BSD was called SunOS. Sun extended the networking tools of the operating system to include the Networked File System (NFS), which was to become an industry Standard. 1993, Sun announced that SunOS, release 4.1.4, would be its last release of an operating system based on BSD. Sun moved to System V, release 4, which

they named Solaris. Sun and AT&T started promoting OPEN LOOK, which they jointly developed. Their goal was to create a consistent look and feel for all flavors of UNIX; unfortunately, OSF had its own GUI called OSF/MOTIF. Thus, round two of the fight for standards began, with MOTIF beating out OPEN LOOK. When MOTIF beat OPEN LOOK in the standards war, Sun conceded, and started to provide a package that contained both OPEN LOOK and MOTIF—called the Common Desktop Environment (CDE)—as standard equipment beginning with Solaris 2.5.1, then after a series of updates, Solaris 7 came out in 1998; Solaris 8 came out in 2001 and Solaris 9 came out in 2002 [37].

5.3 BSD

BSD stands for "Berkeley Software Distribution", the Unix developed at the University of California in Berkeley. Berkeley's Unix was originally derived from AT&T's Unix, but due to legal issues most of AT&T's release was removed and replaced with new code. Eventually, this was ported to the PC in the form of 386BSD, which is what FreeBSD was based on [38].

Despite sharing a common ancestry, the BSD family of operating systems provides a number of complete operating systems packages to meet every need. Basically, there are three BSDs that are well known in the BSD family: OpenBSD, FreeBSD and NetBSD.

- ***FreeBSD***: Perhaps what sets FreeBSD apart most is its technical simplicity. The FreeBSD installation program is widely regarded as the simplest Unix installation tool in existence. Further, its third party software system, the Ports Collection, has been modeled by NetBSD and OpenBSD and remains the most powerful application installation tool available. Through simple one-line commands, entire applications are downloaded, integrity checked, built, and installed making system administration amazingly simple [39,40].

- ***NetBSD***: Today, NetBSD's focus lies in providing a stable, multiplatform, research oriented operating system. NetBSD's portability leads it to run on 33 platforms as of January 2001. Even more impressive is the list of hardware including traditional modern server equipment like standard Intel-based PCs, Compaq's Alpha, or Sun Microsystem's SPARC architectures. Older server and workstation class hardware like the Digital Equipment Corporation's VAX hardware, Apple's Macintosh computers based on Motorola's 68000 processor series are also support. But what really sets NetBSD apart is its support for more exotic hardware including Sega's Dreamcast, Cobalt Network's server appliances, and George Scolaro's and Dave Rand's PC532 hobbyist computer [41,42].

- ***OpenBSD***: OpenBSD diverged from NetBSD around the release of NetBSD 1.1 in November of 1995. OpenBSD's first release came a year later when OpenBSD 2.0 was released in October of 1996. OpenBSD quickly began focusing on producing the most secure operating system available. OpenBSD also advanced the state of code auditing. Beginning in 1996, the OpenBSD team began a line-by-line analysis of the entire operating system searching for security holes and potential bugs. UNIX systems have been plagued for decades by the use of fixed-sized buffers. Besides being inconvenient for the programmer, they have lead to numerous security holes like the fingered exploit in 4.2BSD. Other security holes relating to mishandling temporary files are easily caught. OpenBSD's ground breaking audit has also discovered security-related bugs in related operating systems including FreeBSD, NetBSD, and mainstream System V derivatives. [39,40,42,43,44,45,46].

5.4 OS 360

In April 1964, IBM announced OS/360, an operating system developed to support the new generation and architecture of System/360 hardware - hardware capable of supporting both commercial and scientific applications. Prior to System/360, those applications ran on separate lines of hardware [47].

OS/360 included three control program options, delivered in stages beginning in March 1966. The first stage was the simplest -a sequential scheduler called the primary control program (PCP). PCP performed only one task at a time, and ran in 32KB of memory. With PCP, a processor could spend considerable time waiting for I/O. OS/360 was the first operating system to require direct access devices.

Multiprogramming introduced the technique of assigning control of the processor to another task while the first task was waiting for I/O. This technique utilized resources more effectively. Prior to OS/360, even the most dedicated programmers found I/O programming to be a painful process repetitive, inconsistent and error-prone. OS/360 supplied data and telecommunications access methods that simplified the task.

OS/360 was originally developed as a batch operating system. However, users soon asked for interactive capability. In 1971, IBM released the time-sharing option (TSO), which became an integral part of the operating system. TSO used TCAM, a new telecommunications access method that was developed and released at the same time.

5.5 Windows

Over the past two decades, Microsoft Windows® products have evolved from a single, one-size-fits-all desktop operating system into a diverse family of operating systems and mobile technologies. On November 10, 1983, Microsoft announced Microsoft Windows®, an extension of the MS-DOS operating system that would provide a graphical operating environment for PC users. With Windows, the graphical user interface (GUI) era at Microsoft had begun. Here is some brief introduction for part of its major releases [14,48,49].

- Windows 1.0 (1985): The first version of Windows provided a new software environment for developing and running applications that use bitmap displays and mouse pointing devices.

- Windows 3.0 (1990): The third major release of the Windows platform from Microsoft offered improved performance, advanced graphics with 16 colors, and full support of the more powerful Intel 386 processor. A new wave of 386 PCs helped drive the popularity of Windows 3.0, which offered a wide range of useful features and capabilities.

- Windows NT Workstation 3.5 (1993): This release provided the highest degree of protection yet for critical business applications and data. With support for the OpenGL graphics standard, this operating system helped power high-end applications for software development, engineering, financial analysis, scientific, and business-critical tasks.

- Windows 95 (1995): Windows 95 integrated a 32-bit TCP/IP (Transmission Control Protocol/Internet Protocol) stack for built-in Internet support, dial-up networking, and new Plug and Play capabilities that made it easy for users to install hardware and software.

- Windows NT Workstation 4.0 (1996): This upgrade to the Microsoft business desktop operating system brought increased ease of use and simplified management, higher network throughput, and tools for developing and managing intranets. This release included the popular Windows 95 user interface yet provided improved networking support for easier and more secure access to the Internet and corporate intranets.

- Windows 2000 Professional (2000): Workstation 4.0, Windows 2000 Professional was also designed to replace Windows 95, Windows 98, and Windows NT Workstation 4.0 on all business desktops and laptops. Built on top of the proven Windows NT Workstation 4.0 code base, Windows 2000 added major improvements in reliability, ease of use, Internet compatibility, and support for mobile computing.

- Windows XP (2001): With the release of Windows XP in October 2001, Microsoft merged its two Windows operating system lines for consumers and businesses, uniting them around the Windows 2000 code base [49].

5.6 MS-DOS

Known variously as Seattle Computer 86-DOS, IBM Personal Computer DOS, and Zenith Z-DOS, MS-DOS was developed by Seattle Computer Products for its 8086-based computer system. The MS-DOS history is intertwined with the general development of software for 8086-based computers.

In May 1979, Seattle Computer made the first prototype of its 8086 microprocessor card for the S-100 bus. There were brief discussions with Digital Research about using one of Seattle Computer's prototypes to aid in developing CP/M-86, which was to be

ready "soon". Although Seattle Computer was considering using CP/M-86 when it became available (expected no later than the end of 1979), there were only two working prototypes of the 8086 processor card, and it was felt that both were needed in house. Therefore, there wasn't one free for Digital Research.

Microsoft had already started a strong 8086 software-development program. The firm was ready to try the 8086 version of Stand-Alone Disk BASIC, which is a version of its BASIC interpreter with a built-in operating system. During the last two weeks of May 1979, this BASIC was made completely functional using the hardware that Seattle Computer provided for Microsoft. Seattle Computer Products displayed the complete package (8086 running disk BASIC) in New York the first week of June at the 1979 National Computer Conference. (This was the first-ever public display of an 8086 BASIC and of an 8086 processor card for the S-100 bus.)

In the last few days of 1980, a new version of the DOS was released, now known as 86-DOS version 0.3. Seattle Computer passed this new version on to Microsoft, which had bought non-exclusive rights to market 86-DOS and had one customer for it at the time. Also about this time, Digital Research released the first copies of CP/M-86. In April 1981, Seattle Computer Products released 86-DOS version 1.00, which was very similar to the versions of MS-DOS that are widely distributed today.

In July 1981, Microsoft bought all rights to the DOS from Seattle Computer, and the name MS-DOS was adopted. Shortly afterward, IBM announced the Personal Computer, using as its operating system what was essentially Seattle Computer's 86-DOS 1.14. Microsoft has been continuously improving the DOS, providing version 1.24 to IBM (as IBM's version 1.1) with MS-DOS version 1.25 as the general release to all MS-

DOS customers in March 1982. Now version 2.0, released in February 1983, has just been announced with IBM's new XT computer [50].

5.7 MAC OS

Macintosh OS X, 9, OS 8, OS 7 and OS 6 are desktop operating systems made by Apple Computer that run on Motorola/IBM PowerPC and Motorola 680x0.

In 1987, Apple introduced the Mac II. Built with expandability in mind, the Mac II made the Macintosh line a viable, powerful family of computers. Apple was a "Wall Street darling" again, shipping 50,000 Macs a month. It seemed in 1989 that Windows would be a flop, and the Mac would be riding high for the next decade.

But it didn't. By 1990 the market was saturated with PC-clones of every conceivable configuration, and Apple was the only company selling Macs. In late May, Microsoft rolled out Windows 3.0, which could run on virtually all of the PC-clones in the world. Apple was in trouble.

In 1994 Apple announced the PowerMac family, the first Macs to be based on the PowerPC chip, an extremely fast processor co-developed with IBM and Motorola. The PowerPC processor allowed Macs to compete with, and in many cases surpass, the speed of Intel's newer processors.

In late December 1996, Apple made an industry-shattering announcement that it would be acquiring NeXT, and that Steven Jobs would be returning to the fold. The merger was brought about in order to acquire NeXTstep, which was to become the basis for Apple's next-generation OS, Rhapsody, which was slated for a 1998 release [51,52].

5.8 Linux

Linux is an operating system that was initially created as a hobby by a young student, Linus Torvalds, at the University of Helsinki in Finland. Linus had an interest in Minix, a small UNIX system, and decided to develop a system that exceeded the Minix standards. He began his work in 1991 when he released version 0.02 and worked steadily until 1994 when version 1.0 of the Linux Kernel was released. The kernel, at the heart of all Linux systems, is developed and released under the GNU General Public License and its source code is freely available to everyone. It is this kernel that forms the base around which a Linux operating system is developed. There are now literally hundreds of companies and organizations and an equal number of individuals that have released their own versions of operating systems based on the Linux kernel. The current full-featured version is 2.6 (released December 2003) and development continues.

Apart from the fact that it's freely distributed, Linux's functionality, adaptability and robustness, has made it the main alternative for proprietary Unix and Microsoft operating systems. IBM, Hewlett-Packard and other giants of the computing world have embraced Linux and support its ongoing development. More than a decade after its initial release, Linux is being adopted worldwide as a server platform primarily. Its use as a home and office desktop operating system is also on the rise. The operating system can also be incorporated directly into microchips in a process called "embedding" and is increasingly being used this way in appliances and devices.

Linus didn't want to use Windows and searched for an inexpensive alternative that would run on low cost IBM PC clones. The GNU open source project was progressing very slowly because of political infighting and an attempt to make the same operating

system run the same on numerous processors. Linus received permission to use MINIX as the foundation for his own efforts. MINIX was a small version of UNIX created by Andrew S. Tanenbaum to provide college students with a working version of UNIX with no AT&T owned source code. Linus opened a web site on his university student account and started posting free copies of his source code. During the early days of the project, Linus was posting updated versions several times a day, which directly contradicted the commercial approach of only releasing new versions on an infrequent basis after extensive testing. With the help of a growing number of volunteers (literally tens of thousands), Linus quickly replaced all of MINIX with new all new source code. As Linux caught on in popularity (because it allowed college students and hobbyists to experiment with very cheap Intel hardware), other groups of volunteers ported Linux to a wide variety of additional processors. The success of Linux proved the viability of open source software projects and Linus's approach of rapid and continual incremental updates proved to be an effective method for harnessing volunteer effort and an excellent method for widespread testing on a wide variety of hardware.

Linux has achieved a measure of success. In only a few years, the program has evolved from a hacker's toy into software that is, at least in part, technically superior to Windows NT [53,54,55,56,57,58,59].

5.9 NetWare

NetWare was developed by Novell Data Systems in the late 1970s. Using CP/M (Control Program for Microprocessors, the operating system of choice before the PC solidified DOS) and Unix as the guidelines, a multiuser microcomputer was being built. This was a

typical time-sharing system, with dumb terminals attached by serial cables to a central box containing the CPU, disk, memory, and printer attachments.

With NetWare 4.10, the tools that network managers asked for, and more, were included. The network manager now had complete control over the NDS tree, able to prune, graft, split, and merge sections of the tree from the graphical NetWare Administrator program. With the Simplified and Custom Installation options, new networks were installed and running in about 10 minutes of hands-on work. Novell set the pricing to match NetWare 3.12, making smaller companies that avoided NetWare 4 because they didn't feel they needed NDS at a premium happy to try it on a "free" basis. Add the extra disk space provided by the file compression and other storage enhancements, and NetWare 4.10 actually cost less than a comparable NetWare 3.12 system.

For NetWare 5, the biggest change was with TCP/IP support. Yes, Novell had been talking about it for a long time, and, yes, it had supported TCP/IP to varying degrees for years, but, finally, with version 5 it had true support for TCP/IP. In the past, all requests of the file server were submitted in IPX. True, there may have been a TCP/IP connection to the server, but inside each TCP/IP packet was an IPX packet (a technology called tunneling).

Today, NetWare 5.1 turns a NetWare server into a complete e-commerce powerhouse, including all the server software necessary for full Web site building. Management through a browser interface is part of this package. People may claim that Windows NT or 2000 is a better application server than NetWare, but not if they honestly examine all the software inside the red box [60].

5.10 HP-UX

HP-UX is a UNIX-based operating system made by Hewlett-Packard that runs on HP PA RISC. It was first released in year 1986. Now there are two current versions: 11.11 (aka 11iv1.0) for PA-RISC based hardware and 11.20 (aka 11i v1.5) for Intel Itanium Processor Family (IPF) based hardwaree129. HP-UX support server/mainframe for small to large scale servers. It also has databases server support and mainframe support. HP-UX started earlier than 1986 on their hp9000/500 family, with the HP Focus CPU, it was a multi CPU system, up to 7 CPUs in one box. Then came the HP9000/300 family, these where workstations, also running HP-UX. These where built on the Motorola 680X0 CPU. After that HP introduced the HP9000/400 family also called Apollo, since they merged with them. At this time the PA-RISC based HP-UX came along. The 300 and 400 family was supported up to HP-UX 9.10, this included some of the new things in HP-UX 10.X.

The HP-UX operating system is based on UNIX System V Release 2, with important features from Berkeley Software Distribution 4.2. It also incorporates features of subsequent System V and BSD releases, as well as HP extensions and enhancements. Other contributors to HP-UX include Open Software Foundation, Inc. (OSF$^{(TM)}$), Carnegie-Mellon University, Cornell University, Massachusetts Institute of Technology, and numerous other commercial and educational firms and institutions. In short, HP-UX is essentially an AT&T-type of UNIX with numerous extensions [61,62].

5.11 GNU Hurd

The GNU Hurd is the GNU project's replacement for the UNIX kernel. "Hurd" stands for "Hird of Unix-Replacing Daemons". And, then, "Hird" stands for "Hurd of Interfaces Representing Depth". The Hurd is a collection of servers that run on the Mach microkernel to implement file systems, network protocols, file access control, and other features that are implemented by the Unix kernel or similar kernels (such as Linux).

Currently, the Hurd runs on IA32 machines. The Hurd should, and probably will, be ported to other hardware architectures or other microkernels in the future.

The GNU system (also called GNU/Hurd) is completely self-contained (you can compile all parts of it using GNU itself). You can run several instances of the Hurd in parallel, and debug even critical servers in one Hurd instance with gdb running on another Hurd instance. You can run the X window system, applications that use it, and advanced server applications like the Apache web server.

On the negative side, the support for character devices (like sound cards) and other hardware is mostly missing. Although the POSIX interface is provided, some additional interfaces like POSIX shared memory or semaphores are still under development. All this applies to the current development version, and not to the last release (0.2) [63,64].

5.12 IBM AIX

AIX is based on UNIX System V and Berkeley Software Distribution 4.3 but is more of a hybrid of these two types of UNIX than HP-UX. AIX conforms to the Portable Operating System Interface for Computer Environments (POSIX) and to OSF. It also contains several IBM-proprietary features, such as the Object Data Manager (ODM) and System

Resource Controller (SRC). Its windowing system, AIXwindows Environment/6000 is based on the X Window System with OSF/Motif and is an optional product.

The success of AIX 5L as the native platform for IBM pSeries continues to attract extensive support from the UNIX industry's most successful application vendors. In addition, the affinity features of AIX with Linux provide customers the flexibility to leverage AIX and Linux together where appropriate, helping to preserve investments in skills and applications [65,66,67].

5.13 Compaq/DEC VMS

VMS is a high performance operating system made by DEC that runs on DEC VAX. OpenVMS is an updated version of VMS and runs on both the DEC VAX and the DEC Alpha.

VMS and OpenVMS are the same operating system under two different names (the name changed to OpenVMS about the time POSIX support and a few other "open" items were added.)

OpenVMS, originally called VMS (Virtual Memory System), was first conceived in 1976 as a new operating system for Digital's new, 32-bit, virtual memory line of computers, eventually named VAX (Virtual Address eXtension). The first VAX model, the 11/780, was code-named "Star", hence the code name for the VMS operating system, "Starlet", a name that remains to this day the name for the system library files (STARLET.OLB, etc.). VMS version X0.5 was the first released to customers, in support of the hardware beta test of the VAX-11/780, in 1977. VAX/VMS Version V1.0 shipped in 1978, along with the first revenue-ship 11/780s.

OpenVMS is a 32-bit, multitasking, multiprocessing virtual memory operating system. Current implementations run on Digital's VAX and Alpha computer systems [68,69,70].

5.14 Multics

Multics (Multiplexed Information and Computing Service) is a mainframe timesharing operating system begun in 1965 and used until 2000. Multics began as a research project and was an important influence on operating system development. The system became a commercial product sold by Honeywell to education, government, and industry. This web site describes the hardware, software, and people that made the system the best thing of its kind for many years.

The Compatible Timesharing System (CTSS) was one of the first timesharing systems. It was developed at the MIT Computation Center in 1961 on an IBM 709. In November 1962, MIT Prof. Robert M. Fano leaded the development of MAC (Multiple Access Computers) Bell Labs decided to buy a GE-645 in early 1965 and joined the software development effort, and GE also agreed to contribute. The three organizations worked out a structure for cooperation.

GE sold its computer business to Honeywell in 1970. The last Multics system running, the Canadian Department of National Defence Multics site in Halifax, Nova Scotia, Canada, shut down October 30, 2000 at 17:08Z. This system was modified to be Y2K compliant and was the main production system until Sept/00 [71,72].

5.15 OS/2

A long time ago, IBM and Microsoft still were great pals. At first, Microsoft developed DOS for IBM PC's, and later on, the company started - again by order of Big Blue - the development of OS/2, or Operating System 2. Microsoft and IBM split up a long time ago. That was because Microsoft started to develop its own graphical operating system: Windows and that during the development of the joint-venture between IBM and Microsoft. IBM felt deceived and decided to continue development of OS/2 itself. Debugging and rewriting large portions of OS/2's source code led to a new 32 bit operating system that was rock-stable.

When both operating systems appeared on the market, Microsoft and IBM faced each other. With the popularity of Windows and OS/2, the competition between the two former pals grew. The final collision had place in 1995. OS/2 Warp 3.0, a powerful upgrade of OS/2, had just appeared, and after some months, Microsoft introduced its Windows 95. Though OS/2 had been released some months earlier, and seemed extremely fast and stable, most PC users decided to wait to see which way the cat jumps. Next, Windows 95 was released with such a great advertising campaign that mankind spontaneously forgot about the existence of other alternative operating systems. In 1996, Warp 4.0 (aka Merlin) was released, but in fact, Microsoft had already won [15,73].

After reviewing the brief history and system features of each operating system, how to collect data for intrinsic factors and extrinsic factors will be discussed and the results of the collection will be presented.

CHAPTER 6

DATA COLLECTION

6.1 Resources of the data

In Chapter 4, the methods of how to quantify intrinsic factors and extrinsic factors have been defined. In this chapter, how to collect data based on these rules will be discussed.

Data collection is a hard and time-consuming job. The operating system trends group has used different methods to collect data for the answers:

1. Books: Text books, authorial books about operating system are the first resources. Most of these books have discussed part of the intrinsic factors in the contents. For example, the famous dinosaur book "Operating system concepts" by Abraham Silberschatz, el is a general text book about operating systems. And there are a lot of books especially for a particular operating system. For example, "Mac OS X" by David Pogue is just for Mac OS, "Microsoft Windows XP inside out by" Ed Bott; el. is special for Windows XP. These books provide high level introduction to operating systems.

2. System manuals and handbooks: Every operating system has its own system manual and hand books for user. They provide all the detailed technical information about the system from installation, configuration to all high level technical details for the operating systems.

3. Journals, papers and other articles: There are a lot of papers and articles published about operating system. Actually, operating system is always among the hottest topics for most of the computer technical international conferences, journals and magazines, for example IEEE, ACM, to name but a few. These excellent articles and papers are very good reference.

4. Internet: It is well known that Internet is big resource. There are thousands of thousand of websites, links that are talking about operating systems. Most of the famous organizations have their online website and provide services sometimes more quick and more detail than any other format. Almost for every operating system, there are forum, groups, online organizations, standard, etc that are actively online. These web pages provide the newest information for the operating system research.

5. Surveys: Some of the factors have limited resources or the data obtained from the above sources can not be used. For example, the extrinsic factor of Grassroots Support,

it is difficult to know how many people who prefer to use a particular operating system. In order to get these kinds of information, a survey web page is set up. To store all of the data, a data warehouse is established in software engineering lab in New Jersey Institute of Technology. For the details, visit:
http://swlab.njit.edu/OS/survey.htm

By analyzing the data in this data ware house, good understand could be gained in historical trends of each operating system.

6.2 Survey Webpage

A survey webpage is set up to collect data for some intrinsic and extrinsic factors. By asking survey users to answer the questions designed for each factor, the information about the factor is gathered. Because different questions are designed for different groups of people, the survey is divided into six categories. Survey users are directed to a category of survey questions according to their background.

The six categories are as such: basic survey; feature survey; institutional survey; industrial survey; governmental survey; design survey.

Institutional survey, industrial survey and governmental survey are for *institutional support*, *industrial support* and *governmental support* respectively. They are taken by institutional survey users, industrial survey users and governmental survey users correspondingly. The sub-features of these three factors are listed for survey user to answer for different periods of time.

For example, in institutional survey, users from institutional units are going to answer the following questions for each of the sub-features:

- Whether your institutional unit supports any of the following operating system in each of the following specified years?

- Whether in your institutional unit, you ever teach/take a course using any of the following operating system in each of the following specified years?
- Whether in your institutional unit, you ever teach/take a course or teaching any of the following operating system in each of the following specified years?
- Whether in your institutional unit, you do research by using any of the following operating system in each of the following specified years?
- Whether in your institutional unit, you do research on any of the following operating system in each of the following specified years?

Basic survey includes questions for *grassroots support*. Unlike the above three supports, grassroots support can be applied to all survey users. Therefore, all users from the above surveys are directed to this category. In this survey, users need to answer the three questions for grassroots support for each specified year.

Feature survey contains questions to be answered for *ease of learning, ease of use* and *reliability*. For *ease of learning*, survey users are asked to enter the approximate time for them to learn an operating system. For *ease of use*, all the sub-features are listed. For each of the sub-feature, based on user's experience for this sub-feature, user can choose one of the five ratings from "excellent" to "not good" for a given operating system. For *reliability*, users are asked to provide the number of system failures per month when they use an operating system.

Design survey is used particularly for *design* factors. In this survey, all the sub-features of *design* are listed. For each of the sub-feature, user is asked to choose whether the feature is "excellent", "very good", "good", "acceptable" or "not good" for a given operating system.

All information gathered from the survey webpage is stored in a data warehouse. The survey webpage is open publicly on the Internet. All NJIT students, faculties and

staff, various interested parties are welcome to participate in the survey. Besides, survey invitations are posted to public websites as well as user groups, comp.os.linux, comp.os.ms.windows, comp.os.research, comp.os.unix.misc, comp.os.mach, etc. These operating system related sites are chosen, because they have considerable traffic and people on these websites are balanced. Therefore, it is not favorable to any operating system and thus the results are trustable. In total, up until May 2004, about 800 records are gathered from the survey websites. Figure 6.1 is a screen shot of the web survey page. Also, the research is demonstrated to high school and college students and invited them to participate in the web survey. In order to attract more users, several announcement messages are posted to NJIT web sites, inviting students to participate in the survey.

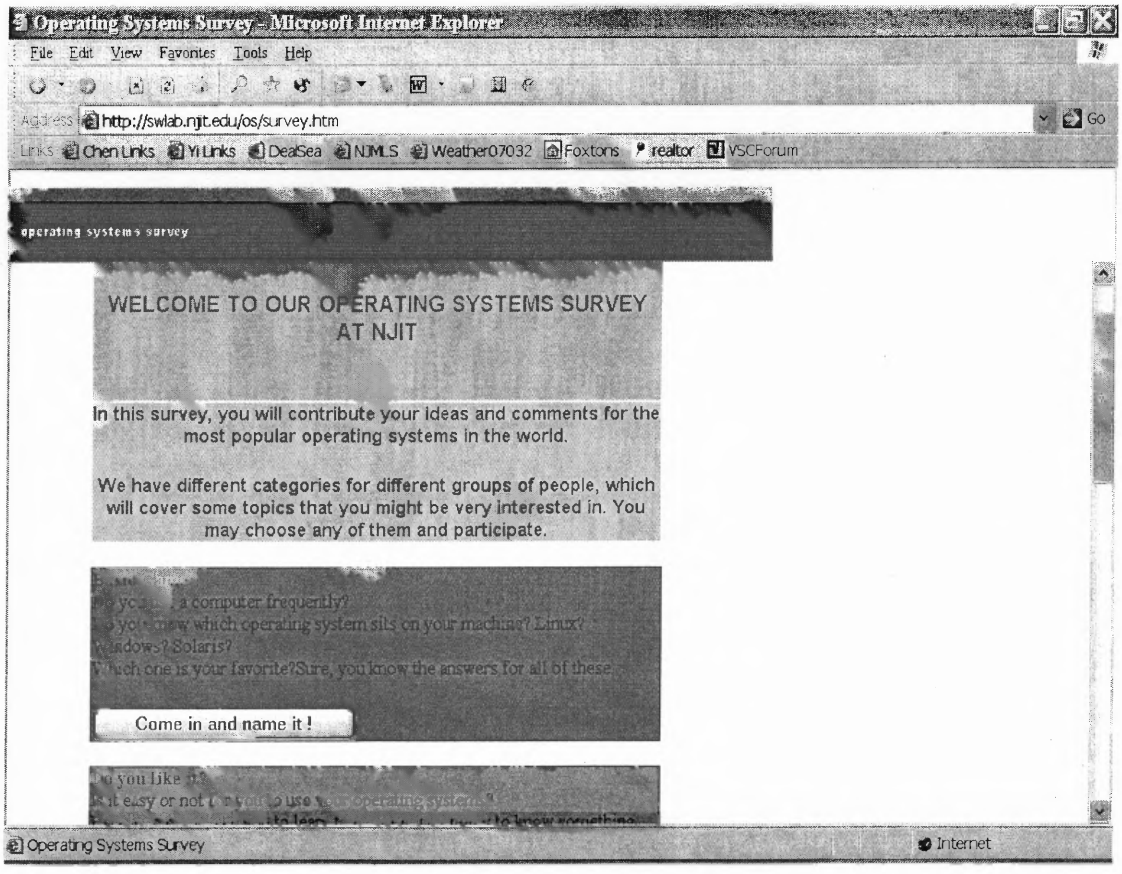


Figure 6.1 Operating System Survey Website.

6.3 Data Collection for Intrinsic Factors

To collect data for intrinsic factors, the operating system evolution research group uses all the methods mentioned in previous sections and tries to evaluate intrinsic factors based on all of these resources.

1. Scalability. All of the above resources are used except the survey web pages to get the data for scalability. By going through all the resources, for each operating system, which sub-factors are supported is decided. Because the quantifying method of scalability is hierarchical sub-feature, the scores for all the supported sub-factors are summarized and the final grade for scalability is derived. Following references are used: [15,24,36,43,46,56,57,58,65,73,74,75,76,77,78,79,80,81,82,83,84,85].

Table 6.1 Scores for Scalability, CPU Management and I/O Management

OS	Scalability	CPU Management	I/O Management
UNIX	28	36	21
Solaris/Sun OS	28	36	21
BSDs	21	28	21
OS/2	15	6	15
Windows	28	28	21
MS-DOS	6	1	3
MAC OS	21	28	21
Linux	28	36	21
NetWare	15	21	15
HP-UX	21	28	21
GNU Hurd	21	28	15
IBM AIX	28	36	21
Compaq/DEC VMS	21	21	21
Multics	10	15	10
OS360	15	21	9

2. CPU Management. Similar as scalability, all of the above resources are used except the survey web pages to get the data for scalability. By going through all the resource, for each operating system, which sub-factors are supported are decided. The quantifying

method of CPU Management is hierarchical sub-feature. The scores for all the supported sub-factors are summarized and the final grade for CPU Management is derived. References are [15,24,36,43,46,56,57,58,65,73,74,75,76,77,78,79,80,81,82,83,84,85].

Table 6.2 Scores for Memory Management

			Memory Allocation Strategies				Virtual Memory					
Operating Systems	**Score**	**Garbage collection**	**Fixed-partition memory strategy**	**Variable partition memory strategy**	**Contemporary allocation strategy**	**Runtime bound checking**	**Swapping**	**Address translation**	**Static paging**	**Dynamic paging**	**Segmentation**	**Shared-memory multiprocessor**
UNIX	29	1	1	2	3	4	1	2		4	5	6
Solaris/Sun OS	30	1	1	2	3	4	1		3	4	5	6
BSDs	29	0	1	2	3	4	1		3	4	5	6
OS/2	16	0	1	2	3	4	1	2	3			
Windows	32	1	1	2	3	4	1	2	3	4	5	6
MS-DOS	7	0	1	2		4						
MAC OS	32	1	1	2	3	4	1	2	3	4	5	6
Linux	32	1	1	2	3	4	1	2	3	4	5	6
NetWare	22	0	1	2	3	4	1	2		4	5	
HP-UX	31	0	1	2	3	4	1	2	3	4	5	6
GNU Hurd	25	0	1	2	3	4	1	2	3	4	5	
IBM AIX	32	1	1	2	3	4	1	2	3	4	5	6
Compaq/DEC VMS	32	1	1	2	3	4	1	2	3	4	5	6
Multics	13	0	1	2		4	1	2	3			
OS360	10	0	1	2	3	4						

3. Memory Management. Recourse used for memory management is all the first 4 resources mentioned above. After deciding which sub-factors are supported by a particular operating system, using hierarchical sub-features quantifying method to get the final score of memory management. The references are listed as following.

[15,24,36,43,46,56,57,58,65,73,74,75,76,77,78,79,80,81,82,83,84,85].

4. I/O Management. Recourse used for I/O management is all the non-survey resources mentioned above. After deciding which sub-factors are supported by a particular

operating system, using hierarchical sub-features quantifying method to get the final score of I/O management. The references include:

[15,24,36,43,46,56,57,58,65,73,74,75,76,77,78,79,80,81,82,83,84,85].

5. Ease of Learning. The score of ease of learning are derived from the webpage. Users of survey are asked to enter the approximate time (number of hours) for them to learn the operating system. Then the score of a particular operating system is the average number. The scores of ease of learning in Table 6.3 are derived from about 500 records.

6. Ease of Use. Same as ease of learning, the score for ease of use is derived from the survey webpage. All 8 sub-features of ease of use are listed and have the discrete scale from "excellent" to "not good" for survey user to choose. When user chose one of the choices, the score for its corresponding sub-features are stored in the data warehouse. After a user evaluates all the sub-features, the user's score for this given operating system is the average value of all the sub-features. And the final score of this operating system is the average value of all users grading. The scores of ease of use in Table 6.3 are derived from about 500 records.

Table 6.3 Scores for Ease of Learning and Ease of Use

OS	Ease of Learning	Ease of Use
UNIX	119.62	3.01
Solaris/Sun OS	121.36	3.98
BSDs	117.37	4.01
OS/2	119.90	3.78
Windows	102.25	4.89
MS-DOS	186.57	2.89
MAC OS	110.38	4.67
Linux	101.01	4.96
NetWare	185.87	2.65
HP-UX	116.01	3.91
IBM AIX	118.76	4.02

7. Consistency of Interaction Protocol. Similar as scalability, all the non-survey resources are used to gather information for consistency of interaction protocol and decide which protocols are supported by a particular operating system. The final score of a given operating system is the number of protocols that is supported by this operating system. The following references are used: [24,36,56,58,74,75,77,85,86].

Table 6.4 Scores for Consistency of Interaction Protocol, System Services and Range of Programming Languages Supported

OS	# of Consistent Interactive Protocol	System Services	Range of Programming Languages Supported
UNIX	10	47	10
Solaris/Sun OS	12	51	10
BSDs	10	51	10
OS/2	8	51	9
Windows	12	51	10
MS-DOS	9	41	8
MAC OS	12	51	10
Linux	12	51	10
NetWare	10	51	10
HP-UX	12	51	10
GNU Hurd	8	40	9
IBM AIX	12	51	10
Compaq/DEC VMS	12	51	10
Multics	8	40	4
OS360	8	40	4

8. System Services. All the non-survey resources are used to gather information for system services and to decide which system services are provided by a particular operating system. Hierarchical sub-factors quantifying method is used to calculate the corresponding score. The following references are used:

[15,24,36,43,46,56,57,58,65,73,74,75,76,77,78,79,80,81,82,83,84,85].

9. Range of Programming Languages. All the non-survey resources are used to gather information for range of programming languages and to decide which programming

languages are supported by a particular operating system. The final score of range of programming languages for a given operating system is the number of programming languages that are supported. The following references are used:

[14,22,87,88,89,90,91,92,93,94,95,96,97,98].

10. Distributed Computing. All the non-survey resources are used to gather information for distributed computing and to decide which distributed computing sub-feature are provided by a particular operating system. Hierarchical sub-factors quantifying method is used to calculate the corresponding score. The following references are used:

[15,23,36,46,64,65,72,74,75,77,78,79,80,82,83,84,99,100,101,102].

Table 6.5 Scores for Distributed Computing, Network Services and Deadlock Management

OS	Distributed Computing	Network Services	Deadlock Management
UNIX	28	22	8
Solaris/Sun OS	28	28	8
BSDs	28	28	8
OS/2	6	15	5
Windows	28	28	8
MS-DOS	3	15	3
MAC OS	28	21	5
Linux	28	28	8
NetWare	28	13	5
HP-UX	28	24	8
GNU Hurd	28	15	5
IBM AIX	28	28	8
Compaq/DEC VMS	28	24	5
Multics	6	6	3
OS360	6	6	3

11. Network Services. All the non-survey resources are used to gather information for network services and to decide which network services are provided by a particular operating system. Hierarchical sub-factors quantifying method is used to calculate the corresponding score. The following references are used:

[15,23,36,46,64,65,72,74,75,77,78,79,80,82,83,84,99,100,101,102].

12. Deadlock Management. All the non-survey resources are used to gather information for deadlock management and to decide which deadlock management strategies are provided by a particular operating system. Hierarchical sub-factors quantifying method is used to calculate the corresponding score. The following references are used: [15,23,36,64,65,72,74,75,77,78,79,80,82,83,84,99,100,101,102].

Table 6.6 Scores for Reliability, Security and Protection Management

OS	Reliability	Security and Protection
UNIX	2.88	45
Solaris/Sun OS	3.12	45
BSDs	2.67	45
OS/2	11.21	29
Windows	12.03	45
MS-DOS	31.13	6
MAC OS	20.13	41
Linux	10.35	45
NetWare	20.13	45
HP-UX	4.31	45
GNU Hurd	9	36
IBM AIX	2.15	45
Compaq/DEC VMS	50	39
Multics	60.1	15
OS360	59.8	15

13. Reliability. The score of reliability are derived from the webpage. Users of survey are asked to enter the approximate number of failure per month for the operating system. Then the score of a particular operating system is the average value. The scores of reliability in Table 6.6 are derived from about 500 records that are gathered from the survey page.

14. Security and Protection Management. All the non-survey resources are used to gather information for security/protection management and to decide which security/protection management strategies that are discussed before are provided by a particular operating

system. Hierarchical sub-factors quantifying method is used to calculate the corresponding score. The following references are used:

[15,23,36,64,65,72,74,75,77,78,79,80,82,83,84,99,100,101,102].

15. Portability. All the non-survey resources are used to gather information for portability and to decide how much effort (number of man-month) to transfer the operating system from one hardware or software environment to another. Straightforward numeric formula quantifying method is used to calculate the corresponding score. The average value of all the numbers obtained is the final score. The following references are used:

[24,41,79,103,104,105,106,107,108,109,110,111,112].

Table 6.7 Scores for Portability, Compatibility and Openness

OS	Portability Average Number of Man-month	Compatibility Average Number of Man-month	Openness
UNIX	0.5	4	47
Solaris/Sun OS	∞	4	47
BSDs	0.5	2	35
OS/2	0.5	2	27
Windows	0.5	2	30
MS-DOS	0.5	0.5	1
MAC OS	∞	1	41
Linux	0.5	2	40
NetWare	1	1	22
HP-UX	∞	6	46
GNU Hurd	0.5	1	36
IBM AIX	∞	6	47
Compaq/DEC VMS	∞	1	46
Multics	∞	4	0
OS360	∞	4	0

16. Compatibility. All the non-survey resources are used to gather information for compatibility and to decide how much effort (number of man-month) to upgrade or downgrade an operating system. Straightforward numeric formula quantifying method is

used to calculate the corresponding score. The average value of all the numbers obtained is the final score. The following references are used:

[24,41,79,103,104,105,106,107,108,109,110,111,112].

17. Openness. All the non-survey resources are used to gather information for openness and to decide which standards are complied by a given operating system. Hierarchical Sub-feature quantifying method is used to get the final score for every operating system. The following references are used: [13,30,31,68,113,114,115,116,117].

Table 6.8 Scores for Design

OS	Design Score
UNIX	4.813
Solaris/Sun OS	4.283
BSDs	3.610
OS/2	2.769
Windows	4.747
MS-DOS	1.830
MAC OS	4.036
Linux	4.859
NetWare	2.883
HP-UX	3.547
IBM AIX	4.763
Compaq/DEC VMS	2.865
OS360	1.275

18. Scores for Design. The score of design are derived from the webpage. Every sub-feature of design is listed and has the discrete scale from "excellent" to "not good" for survey user to choose. When user chose one of the choices, the score for its corresponding sub-features are stored in the data warehouse. After user evaluates all the sub-features, the user's score for this given operating system is the average value of all the sub-features. And the final score of this operating system is the average value of all users grading. The scores of design in Table 6.8 are derived from about 400 records.

Table 6.9 Scores for Cost (All prices are in US$ for ease of conversion to your currency, and correct as of 4-19-2002)

OS	Acquisition	Maintenance	Operation
UNIX	$34,500	$105,000	$25,000
Solaris/Sun OS	$27,500	$111,000	$13,420
BSDs	Free		
OS/2	$250		
Windows	$17,300	$148,000	$63,000
MAC OS	$11,672	$1,810	$1,461
Linux	$27	$150,000	$64,000
NetWare	$8,000	$102,615	$60,100
HP-UX	$33,660	$65,340	$21,000
IBM AIX	$14,875	$27,625	$29,500
Compaq/DEC VMS	$7,056		
OS360	$1,328	$854	$1,793

19. Cost. All the non-survey resources are used to gather information for cost. Straightforward numeric formula quantifying method is used to calculate the corresponding score. The average value of all the numbers obtained is the final score. The following references are used: [118,119,120,121,122,123,124].

6.4 Data Collection for Extrinsic Factors

Similar to intrinsic factors, the operating system trends research group uses all the methods mentioned above and tries to evaluate extrinsic factors based on all of these resources. Most of the extrinsic supports have their results from the survey webpage. Some of the extrinsic support, for example, organizational support, data is collected from other resources as in the references. Different from intrinsic factors, extrinsic is historical based. For each of the extrinsic factors, three groups of historical data are collected: 1997, 2000, and 2003. These historical data are used for the trend watch analysis.

6.4.1 Institutional Support

Table 6.10 shows the results for institutional support. As discussed in Chapter 4, there are five different supports for institutional supports: support the OS, teach using the OS, teach the OS, research using the OS and research on the OS, from weakest support (score: 1) to the strongest (score: 5) respectively. One survey instance will result a score from 5 to 0, providing that if a user does not provide information for this OS will results 0 point for this instance. The average score of all the survey results is used for institutional support as the final score for a given year.

Table 6.10 Scores for Institutional Support

OS	Institutional Support Score		
	1997	2000	2003
UNIX	12.3	12.023	12.352
Solaris/Sun OS	13.21	13.25	12.983
BSDs	10.35	10.328	10.324
OS/2	2.04	2.146	1.875
Windows	13.26	13.348	13.452
MAC OS	2.3	2.145	2.035
Linux	9.88	10.214	11.322
NetWare	3.26	3.249	3.245
HP-UX	8.23	8.142	8.251
GNU Hurd	6.22	6.327	6.451
IBM AIX	9.24	9.358	9.458
Compaq/DEC VMS	5.12	5.197	5.214

From Table 6.10, it can be seen that during the time span, UNIX and Solaris/Sun OS are one of the most popular operating systems in academic area. Instructors use UNIX and Solaris/Sun OS as examples more often than most of other operating systems when they teach operating system courses. OS/2, HP-UX, MAC-OS are less widely used and researched by institutional purposes compared to other popular OS. Also, people like to use Windows as a tool in classrooms as well in libraries.

6.4.2 Industrial Support

From Table 6.11, it can be seen that Solaris, IBM AIX and Windows have very strong industrial support. They remain a strong position in winning industrial support throughout the time span. But what's mentioning is that in 1997, Linux was in the second tier in terms of industrial support, however, during the six years period from 1997 to 2003, it has gained considerable momentum in gaining up speed. By 2003, Linux has come up to the second place, which is just behind Windows operating system.

Table 6.11 Scores for Industrial Support

OS	Industrial Support Score		
	1997	2000	2003
UNIX	2.56	2.555	2.605
Solaris/Sun OS	2.605	2.481	2.427
BSDs	1.61	1.551	1.493
OS/2	1.005	0.726	0.484
Windows	2.62	2.663	2.77
MAC OS	1.15	1.127	1.11
Linux	2.12	2.281	2.607
NetWare	1.62	1.656	1.726
HP-UX	1.995	1.979	1.874
GNU Hurd	1.015	1.156	1.157
IBM AIX	2.565	2.573	2.579
Compaq/DEC VMS	2.055	1.825	1.73

6.4.3 Governmental Support

From Table 6.12, it can be seen that Unix, Windows and IBM AIX has considerable governmental support continuously from 1997 and 2003. Though each of them gained some field in winning more governmental support, they stay relatively stable as far as their ranking is concerned. Similar to organizational support, Linux catches up from behind very quickly and gained rather impressive progress in winning more government support.

Table 6.12 Scores for Governmental Support

OS	Governmental Support Score		
	1997	2000	2003
UNIX	2.56	2.574	2.575
Solaris/Sun OS	2.065	2.063	2.061
BSDs	1.61	1.627	1.561
OS/2	0.618	0.515	0.515
Windows	2.617	2.602	2.655
MAC OS	0.516	0.532	0.343
Linux	2.104	2.235	2.735
NetWare	1.052	1.049	1.052
HP-UX	2.285	2.275	2.262
GNU Hurd	0.511	0.512	0.492
IBM AIX	2.318	2.319	2.316
Compaq/DEC VMS	1.579	1.512	1.493

6.4.4 Organizational Support

Table 6.13 shows the results for organizational support. Unlike other extrinsic factors, the data of organizational support are not obtained from survey; instead it's from a lot of other resources as listed in the references. It can be seen that UNIX, Windows, Linux, BSDs are the most popular platform that are used by people in order to publish papers and articles during 1999 through 2000. But in year 2001~2002, most of the number for this category dropped down except Linux. The following references are used: [34,35,99,102,113,116,118,125,126,127,128,129,130,131,132].

6.4.5 Grassroots Support

From the definition of grassroots support in Chapter 4, it refers more to the general audience of operating system. From Table 6.14, it can be seen that Windows and IBM AIX are far ahead as far as Grassroots support is concerned. The relative ranking of grassroots support remains stable.

Table 6.13 Scores for Organizational Support

OS	Organizations	Major International Conferences	Papers/Articles Published on the OS			Papers/Articles Using the OS		
			1997	2000	2003	1997	2000	2003
UNIX	1	7	248	117	86	229	12000	993
Solaris/Sun OS	2	4	14	6	9	22	680	43
BSDs	1	6	22	0	3	14	1890	49
OS/2	3	2	21	1	0	2	61	29
Windows	2	17	225	45	63	18	5500	181
MS-DOS	1	0	27	2	2	11	113	62
MAC OS	2	6	0	2	9	2	11	12
Linux	5	18	177	49	104	27	3500	2579
NetWare	2	4	10	1	3	0	71	7
HP-UX	2	5	5	1	3	0	18	1
GNU Hurd			0	0	0	0	3	0
IBM AIX	2		16	2	3	2	566	26
Compaq/DEC VMS							342	0
Multics			42	0	2	17	679	74

Table 6.14 Scores for Grassroots Support

OS	Grassroots Support Score		
	1997	2000	2003
UNIX	2.516	2.506	2.581
Solaris/Sun OS	1.743	1.782	1.802
BSDs	0.519	0.521	0.524
OS/2	0.826	0.627	0.4925
Windows	2.7115	2.724	2.748
MAC OS	1.732	1.698	1.786
Linux	2.0125	2.1175	2.3925
NetWare	0.476	0.482	0.479
HP-UX	0.17	0.135	0.155
GNU Hurd	0.092	0.101	0.095
IBM AIX	0.223	0.218	0.233
Compaq/DEC VMS	0.112	0.122	0.131

CHAPTER 7

DATA ANALYSIS & MODEL CONSTRUCTION

In the previous chapters, the following questions have been discussed: how to find out the relevant factors, how to quantify those relevant factors, and how to collect data for them. After the data warehouse has been established, the new questions are: how to analyze these data, how to construct the proper statistical models to describe the history of operating systems, and how to extend the statistical models to predict the evolution of operating systems.

7.1 Constructing Statistics Models

In this project, SPSS statistic toolbox [133] will be used to help describe historical operating system trends and predict the evolution of future trends. First, the data collected will be analyzed to extract some descriptive properties to characterize the evolution of operating system over time. Second, the available data will be used to predict the future trends of operating systems.

Before the statistical model is presented, the following premises are considered:

- Intrinsic factors are adopted as independent variables for the model, as they influence the destiny of an operating system.
- Extrinsic factors are adopted as dependent variables in the model; the status of an operating system is not represented by the simple binary term of successful/unsuccessful, as this would be arbitrarily judgmental. Rather, the status of an operating system is represented by the vector of all the extrinsic factors.
- Because many extrinsic factors feed unto themselves (e.g. the more grassroots support an operating system has, the more grassroots support it may have in the

future) and many influence others (e.g. institutional support influences industrial support), past values of extrinsic factors are adopted as independent variables.

Overall, the independent variables of the model include the intrinsic factors and the historical data of extrinsic factors, and the dependent variables include the future values of the extrinsic factors.

The statistical model is built to use principle component analysis (PCA)[134], Pearson correlation analysis in this project to elucidate the relationships between independent variables and dependent variables. After analyzing the inputs, multiple regression is used to predict future evolution of an operating system by feeding it time-independent intrinsic factors, as well as past and present values of the extrinsic factors. Section 7.2 will discuss how to analyze factors, and how to construct models will be discussed in Section 7.3.

7.2 Independent Data Analysis

7.2.1 Factor Analysis

The factor analysis [134] is used to do the startup research on raw data. The independence of the variables in groups is evaluated for further research. The relatively small number of latent factors proved that many variables are highly correlated and need either adjustment or combination, which make it necessary to do canonical analysis.

Factor analysis is a generic term for a family of statistical techniques concerned with the reduction of a set of observable variables in terms of a small number of latent factors. It has been developed primarily for analyzing relationships among a number of measurable entities (such as survey items or test scores). The underlying assumption of

factor analysis is that there exist a number of unobserved latent variables (or "factors") that account for the correlations among observed variables, such that if the latent variables are partially out or held constant, the partial correlations among observed variables all become zero. In other words, the latent factors determine the values of the observed variables. Each observed variable could be expressed as a weighted composite of a set of latent variables. The primary purpose of factor analysis is data reduction and summarization. Factor analysis has been widely used, especially in the behavioral sciences, to assess the construct validity of a test or a scale.

Once the input data are prepared for the analysis, it is necessary to decide on a factoring technique, that is, a method of extracting factors. There are a variety of different methods of factor extraction available in SPSS and PCA statistical analysis methodology is used to identify a small number of factors that explain most of the variance observed in a much larger number of manifest variables. The following goals are to be reached:

- Reduce the number of components.
- The extracted components should preserve most of the relations with the initial factors.

In this project, three sets of extrinsic factors data (1997, 2000, and 2003), and 15 operating systems will be used as observations. So in final model, the construction of each factor is based on 45 observations.

Table 7.1 shows the factor analysis for intrinsic factors. From Table 7.1, six factors could be used to cover 95.903% of the variance. That means that six-factor model could be used to do factor analysis, while keeping most of the information. Hence for all intents and purposes, the six derived components represent a space of dimension six rather than nineteen. Thus, six components from the initial nineteen intrinsic factors are extracted.

Table 7.1 Factor Analysis for Intrinsic Factor Matrix: Total Variance (Extraction Method: Principal Component Analysis.)

Component	Eigenvalues	% of Variance	Cumulative %
1	9.511	55.949	55.949
2	2.435	14.326	70.276
3	1.992	11.719	81.994
4	1.017	5.981	87.975
5	0.983	5.780	93.755
6	0.365	2.148	95.903
7	0.270	1.586	97.489
8	0.239	1.404	98.892
9	0.116	0.680	99.573
10	0.064	0.377	99.950
11	0.008	0.050	100.000
12	0.000	0.000	100.000
13	0.000	0.000	100.000
14	0.000	0.000	100.000
15	0.000	0.000	100.000
16	0.000	0.000	100.000
17	0.000	0.000	100.000

Table 7.2 Rotated Factor Pattern for Intrinsic Factors Component Matrix (Extraction Method: Principal Component Analysis.)

Intrinsic Factors	**Component**					
	1	2	3	4	5	6
CPU	0.746	0.312	0.407	-0.190	0.331	0.026
Memory	0.620	0.202	0.699	0.198	0.004	0.118
Scalability	0.462	0.540	0.527	-0.125	0.261	0.172
IO	0.391	0.354	0.664	0.407	0.178	0.217
Consistency Of Interaction Protocol	0.485	0.215	0.564	0.590	-0.027	-0.090
System Services	-0.040	0.222	-0.007	0.964	0.027	0.041
Range Of Programming Languages	0.700	0.113	0.232	0.637	0.111	0.035
Distributed Computing	0.933	-0.131	0.301	-0.049	0.010	0.047
Network Service	0.340	0.441	0.530	0.330	0.259	0.451
Deadlock	0.395	0.572	0.175	0.134	0.587	0.251
Security Protection	0.883	0.200	0.057	0.296	0.289	0.040
Compatibility	0.064	0.212	0.274	0.149	0.862	-0.258
Openness	0.175	-0.004	0.871	-0.008	0.412	-0.073
Design	0.335	0.676	0.229	0.480	0.240	0.014
Ease of Use	0.018	0.952	0.005	0.070	0.177	-0.025
Reliability	0.170	0.155	0.112	-0.033	0.909	0.233
Ease Of Learning	-0.027	0.824	0.231	0.487	0.092	0.081

From another perspective, by using previous intrinsic factor matrix, the six-factor model could be constructed. Table 7.2 shows the relationships between new independent factors (components) and original intrinsic factors. As it shows, each of the 19 factors is actually covered by at least one of the six refined components. For instance, component 1 is highly related to factor distributed computing (0.933), security protection (0.883), CPU (0.746). Component 4 covers factors of system services (0.964) and range of programming languages (0.637).

Therefore, the extracted 6 components satisfy the criteria listed at the beginning of this section.

For extrinsic factors, the same method will be used. Table 7.3 shows the factor analysis for extrinsic factors. From this table, the following conclusion can be drawn. For the extrinsic factors, three extrinsic factors will cover 98.056% information. Also, an interesting fact shows that five extrinsic factors will cover 100% information. Table 7.4 shows the five-factor models. However since there are only five extrinsic factors, all of them will be used.

7.2.2 Canonical Correlation Analysis

There are several measures of correlation to express the relationship between two or more variables. Canonical Correlation [135] is an additional procedure for assessing the relationship between variables. Specifically, this analysis allows the analysis of the relationship between two sets of variables. If the square root of the eigenvalues is taken, then the resulting numbers can be interpreted as correlation coefficients. Because the correlations pertain to the canonical varieties, they are called Canonical Correlations. Like the eigenvalues, the correlations between successively extracted canonical variants

are smaller and smaller. Therefore, as an overall index of the canonical correlation between two sets of variables, it is customary to report the largest correlation, that is, the one for the first root. However, the other canonical variants can also be correlated in a meaningful and interpretable manner.

Table 7.3 Factor Analysis for Extrinsic Factor Matrix (Extraction Method: Principal Component Analysis.)

Component	Eigenvalues	% of Variance	Cumulative %
1	4.246	84.927	84.927
2	0.420	8.396	93.323
3	0.237	4.732	98.056
4	0.078	1.556	99.611
5	0.019	0.389	100.000

Table 7.4 Rotated Factor Pattern for Extrinsic Factors

	Component 1	Component 2	Component 3
Government	0.967	-0.080	-0.041
Institution	0.906	0.190	-0.371
Industrial	0.965	-0.206	-0.003
Grassroots	0.925	-0.322	0.176
Organization	0.838	0.481	0.257

In this project, Pearson Correlation [134,135] is used to analyze the relationship among all factors. By doing this, the association between several intrinsic factors and one extrinsic factor is observed. Most results show a relationship, which counts for part of the feature. Different intrinsic factors of an operating system do have different impact on the overall performance by using this model. Please check Table 7.5 for the question: "How is the extrinsic factor government support affected by the intrinsic factors?"

Table 7.5 shows that different intrinsic factors have different impact on the extrinsic factor of Organizational support. Among all these intrinsic factors, some of them are more weighted: security protection, scalability, design and network service have more correlation value for this extrinsic factor than other intrinsic factors.

Table 7.5 Sample Correlation Results for Intrinsic Factors Only

Factor	Government Support
Security Protection	0.883
Scalability	0.789
Design	0.750
Network Service	0.747
Deadlock	0.709
IO	0.666
CPU	0.643
Range Of Programming Languages	0.612
Memory	0.589
Compatibility	0.589
Consistency of Interaction Protocol	0.578
Ease of Learning	0.553
Reliability	0.455
Openness	0.426
Ease of Use	0.425
Distributed Computing	0.408
System Services	0.291

7.2.3 Statistics Conclusion

From the previous sections, two aspects of data analysis are discussed.

First, for intrinsic factors, the factor analysis showed that only six extracted components sufficient to contain more than 95% of the operating systems internal information. Consequently, it shows that the nineteen latent intrinsic factors are highly correlated and they are not independent from each other. To construct useful regression model for the historical trends, the independent extracted components, instead of original factors, will be used. From the factor analysis, it can be inferred that the six components

that are derived from the intrinsic factors do represent important features of the operating systems.

The canonical correlation analysis result shows which factors are mostly related and how much are they related. It is also a good startup to construct regression model, because the newly constructed independent factors, instead of the original factors, will be used to construct the regression model.

CHAPTER 8

TOWARDS A PREDICTIVE MODEL

In the previous chapter, factor analysis has been done; the independent variables (including the new six components extracted from the original 19 intrinsic factors and all the 5 extrinsic factors) have been derived. By doing canonical correlation analysis, two correlation models for intrinsic factors only and for all factors have been made to show the correlation among those factors. This chapter will discuss how to construct regression models by using these independent intrinsic and extrinsic factors and analysis results from the constructed model.

8.1 Regression Model

8.1.1 Regression Analyses

Regression analyses [134,135] are a set of statistical techniques that allow one to assess the relationship between one dependent variable (DV) and several independent variables (IVs). Multiple regression method is an extension of bivariate regression in which several independent variables are combined to predict the dependent variable. Regression may be assessed in a variety of manners, two of the common used are as follows:

- *Partial regression and correlation*: Isolates the specific effect of a particular independent variable controlling for the effects of other independent variables. This method will try to find out the relationship between pairs of variables by recognizing the relationship with other variables.

- *Multiple regression and correlation:* Multiple regression and correlation method combines the effect of all the variables acting on the dependent variable; for a net, combined effect. Thus the resulting R2 value provides an indication of the goodness of fit of the model.

In the social and natural sciences multiple regression procedures are very widely used in research. In general, multiple regression allows the researcher to ask (and hopefully answer) the general question "what is the best predictor?". For example, educational researchers might want to learn what the best predictors of success are in high-school [136].

8.1.2 Multiple Regression Model

Therefore, when using multiple regression and correlation it is often the case that a dependent (or response) variable may depend on more than one independent (or explanatory) variable. Based on the requirements and the different usages of the above two dissimilar regression and correlation methodology, multiple regression method are a good choice to analyze the operating systems revolution.

The multiple regression equation is of the form:

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_q X_q + \epsilon$$

Where:

Y = the predicted value on the DV,

α = the Y intercept, the value of Y when all Xs are zero,

X = the various IVs,

β = the various coefficients assigned to the IVs during the regression,

ϵ = an error term.

q = dimensional hyperplane (number of factors)

Accordingly, a different Y value is derived for each different case of independent variable. The goal of the regression is then to derive the β values, the regression coefficients, or beta coefficients. The beta coefficients allow the computation of

reasonable Y values with the regression equation, and provide that calculated values are close to actual measured values.

Computation of the regression coefficients provides two major results:

- Minimization of deviations (residuals) between predicted and obtained Y values for the data set.
- Optimization of the correlation between predicted and obtained Y values for the data set.

As a result, the correlation between the obtained and predicted values for Y relates the strength of the relationship between the dependent variable (DV) and independent variables (IV). Although regression analysis reveals relationships between variables, this does not imply that the relationships are causal. Demonstration of causality is not a statistical problem, but an experimental and logical problem [136].

8.1.3 Regression Model for Historical Trends

When using multiple regression models for this project, the derived regress model is established to analyze the data. Figure 8.1 gives an illustration for the model.

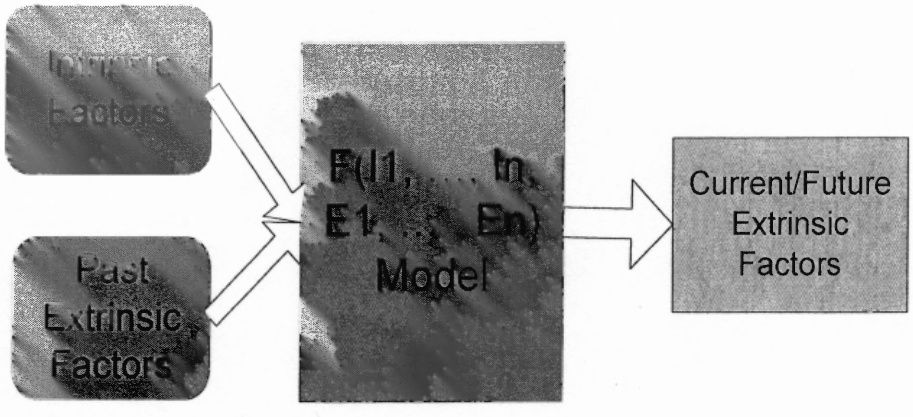
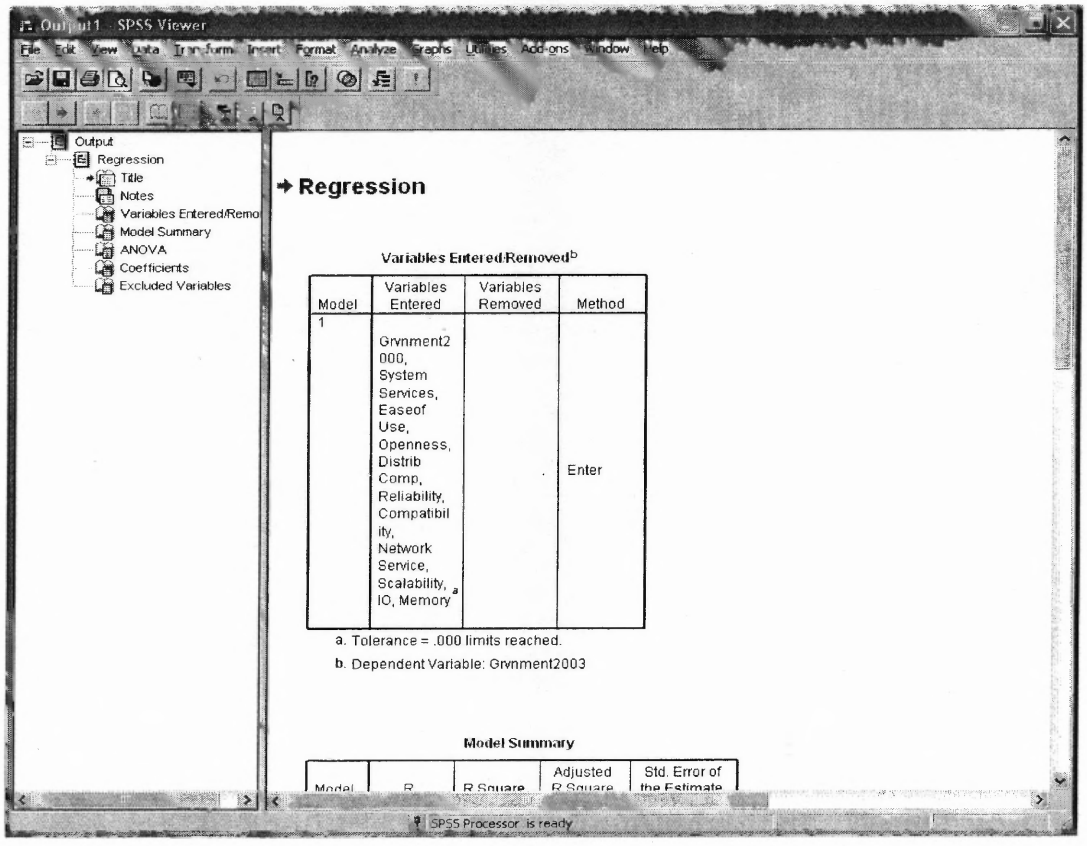


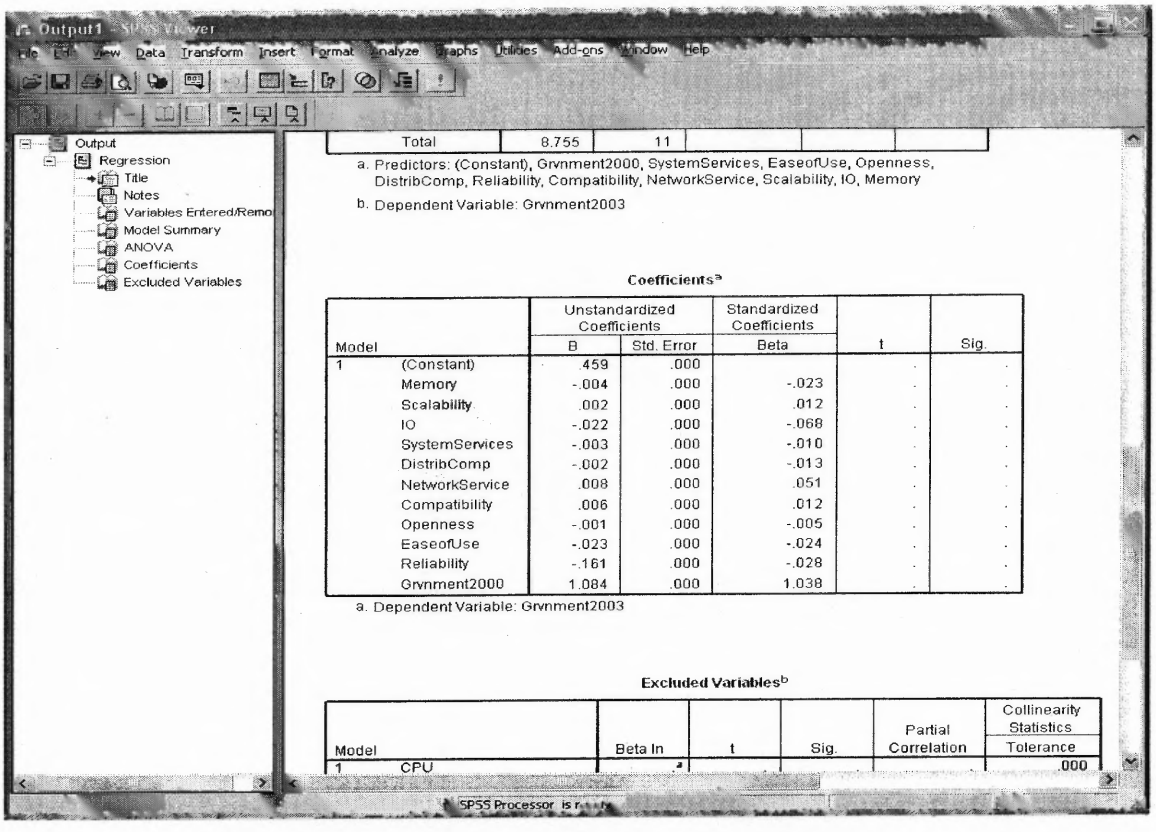
Figure 8.1 Regression Model for Operating System Trend.

In this model, all of the intrinsic factors will be considered as input factors. On the other hand, extrinsic factors actually change across different time period. Thus, the earlier past data will be considered as input factors and the later present and future data will be considered as output factors.

- Input data: intrinsic factors and past extrinsic factors
- Output data: present and future extrinsic factors

The general purpose of multivariate regression is to learn more about the relationship between several independent or predictor variables and a dependent or criterion variable. Multivariate multiple regression can easily be extended to deal with situations where the response consists of $q > 1$ different variables. The input matrix will be constructed by factors obtained from factor analysis; this analysis is necessary to build independent factors so the regression model can be made and interpreted.





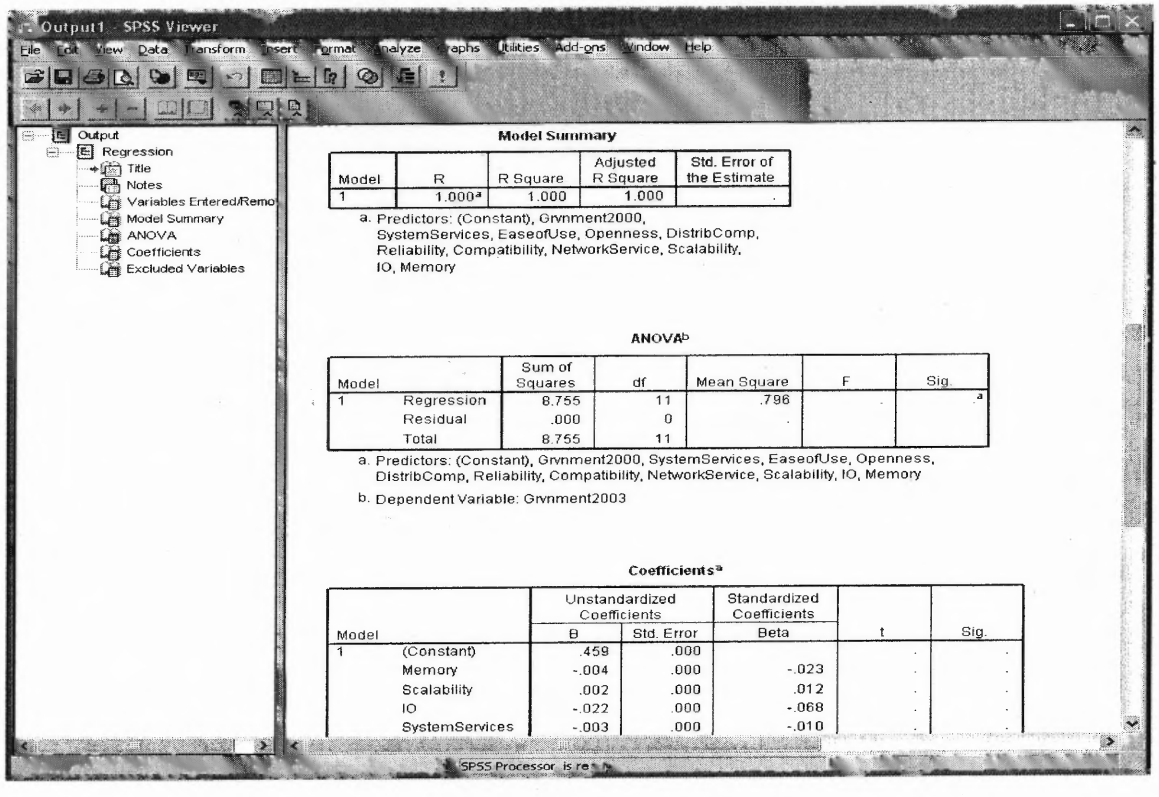


Figure 8.2.A~C Sample SPSS Regression Model Reports for One Extrinsic Factor.

Multiple regression model for operating systems trends consists of 5 parts, one for each dimension. Due to the number of factors used in the model, there will be a balance between model reliability and the information completion. The more factors are used, the more information will be included, but regression model will be less reliable. The fewer factors you used, the less information will be included, but the regression model will be more reliable. The following set of figures (Figure 8.2 A~C) shows an example of the process to create a report based on the SPSS analysis model for extrinsic factor of Governmental Support.

In Figure 8.2, one historical extrinsic factor and 19 independent intrinsic factors are used as input factors against on the multivariate regression model. This report will show the regression model for one extrinsic factor: "How much government support an operating system will receive?" The parameter will show the impact of each input independent factor to the output extrinsic factor. The other regression reports can be found in the website this project uses. By using all of the parameters, the regression models are constructed for historical operating system trends.

8.2 Predictive Model

From the previous section, the derivative prediction models have been constructed for each extrinsic factor in 2006, by submitting the value in 2000 to the 97 position and 03 to the 2000 position in the model.

$$E_{2006} = \alpha * I + \beta * E_{2003} + \gamma * E_{2000} + \epsilon$$

Where:

E_{2006}: Value of extrinsic factors in 2006

I: Value of intrinsic factors

α: Parameter matrix for intrinsic factors

E_{2003}: Value of extrinsic factors in 2003

β: Parameter matrix for extrinsic factors in 2003

E_{2000}: Value of extrinsic factors in 2000

γ: Parameter matrix for extrinsic factors in 2000

ϵ: Constant value

Both intrinsic factors and extrinsic factors are considered in the model. Least square [134,135] is used as the criteria to judge whether the regression model converges. SPSS is used to calculate the parameter matrix so that the least square goal is met. When least square is met, the output with this parameter matrix is the closest to all observations that are fed into the model. Therefore, the model can be used to describe the trends of operating systems.

Some of the predictions are showed in the rest of this chapter. Please check "Trends of Government Support in 2006". Figure 8.3 is the predictive results.

From above graphs, the five operating systems that will receive most of the government support in 2006 will be: Linux, Solaris, UNIX, IBM AIX and Windows. Figure 8.3 shows interesting trends for operating system. It seems like from 2003 to 2006, Linux will overtake windows as the operating systems that will receive the most organizations support. All other operating systems will stay relatively stable in terms of the level of organizational support they receive.

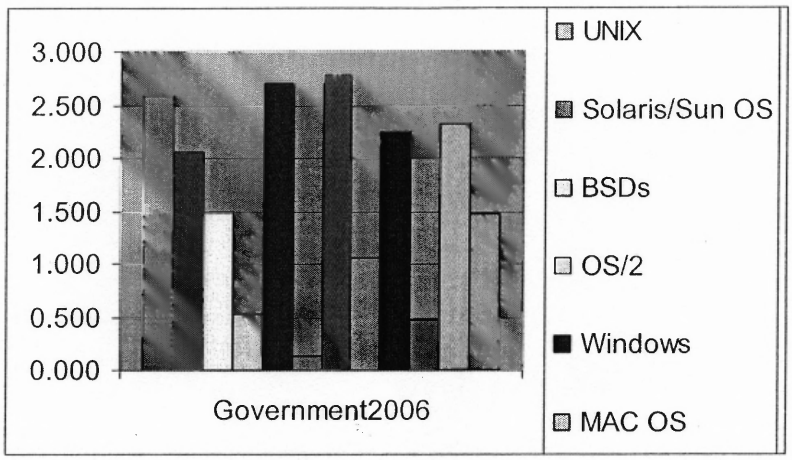


Figure 8.3 Trends of Government Support in 2006.

From another figure (Figure 8.4), it can be seen that Linux, Windows, Solaris and UNIX will still gain more organizational supports than most of the other operating systems. Some operating systems will again remain similar as before, for example, OS/2, GNU Hurd, and HP-UX. Then another question is for a specific operating system, what are the changes for it during these periods of time. The changes of the organizational support from year 1997 to 2006 are shown in Figure 8.5. From this figure, most of the "popular" OS will remain stable even gain more support for the organizational support. For example, UNIX and Solaris maintain quite good position. Linux and Windows are very "healthy" and even jump very fast across the year, especially for Linux. Some OS are losing their organizational support, for example, BSDs, OS/2. Some operating system remains almost same, for example, NetWare, Compaq, etc, which actually could be found in the bottom of the figure.

Notice that this prediction is based on historical trends, which means that if future trends remain the same as current trends, Linux will still keep growing.

Figure 8.6 shows another interesting question for grassroots support: "How the grassroots support is going to change from 1997 to 2006?"

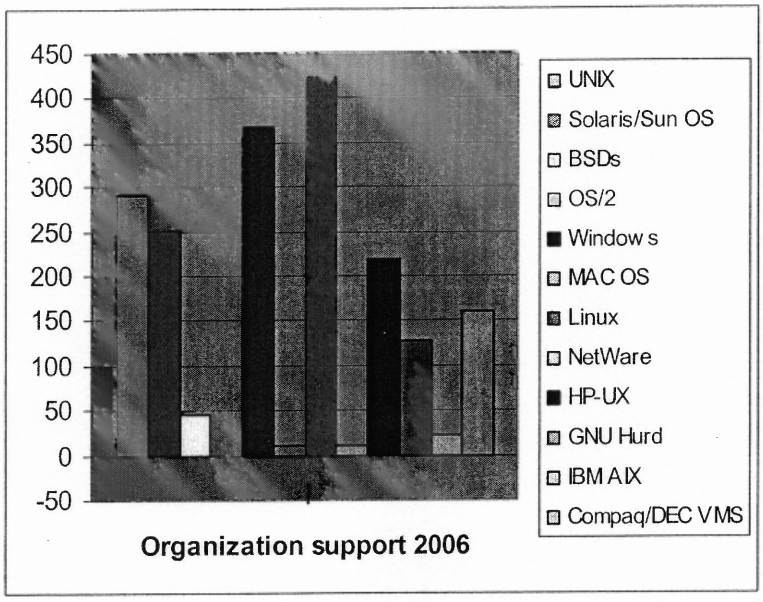


Figure 8.4 Trends of Organization Support in 2006.

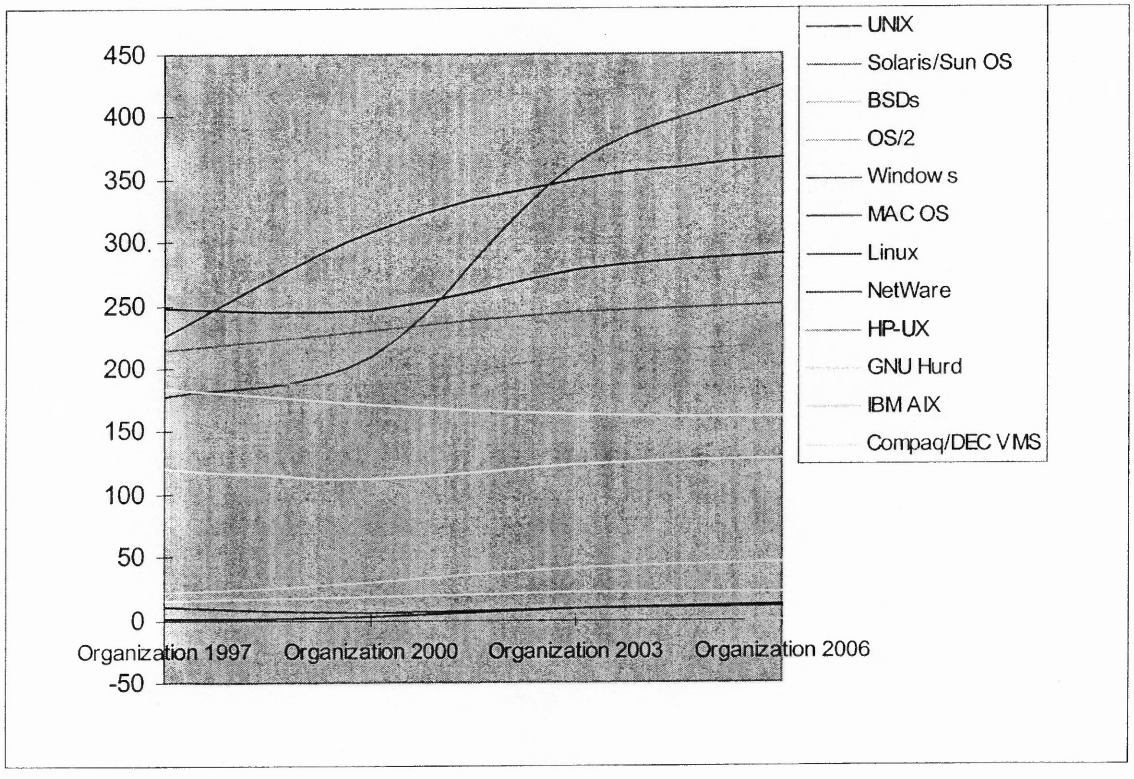


Figure 8.5 Organization Support from 1997 to 2006.

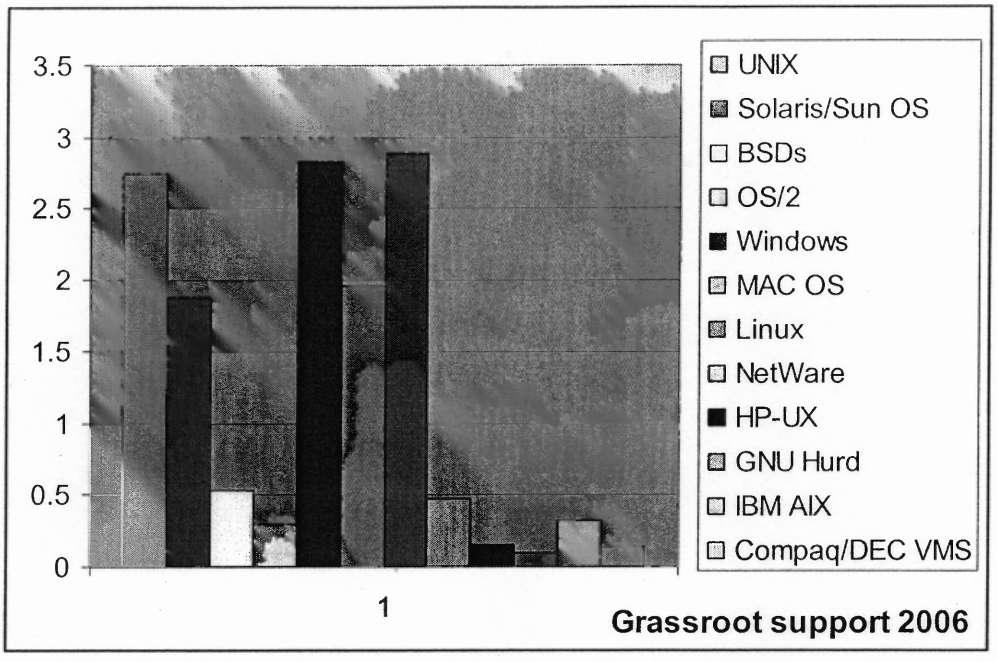


Figure 8.6 Grassroots Support in 2006.

From above graph, the top operating systems that will receive the most grassroots support in 2006 will be: UNIX, Solaris/Sun OS, Windows, Linux, and MAC OS. And some other OS still have their small portion of support, such as NetWare, GNU Hurd, HP-UX, etc. General population's point of view is considered to decide the grassroots supports. Popular PC operating systems such as Windows and Linux gain more support because they are widely used on PC. On the other hand, Unix and Solaris are commonly installed on workstations thus people are familiar with them.

Because the metrics are kept the same when the data is gathered and when the prediction is done so the values are comparable across the whole time period.

Figure 8.7 also shows interesting trends of grassroots support for operating system. From this figure, some of the operating systems stay relatively stable for their support, for example, HP-UX, IBM AIX, NetWare, BSDs. These operating systems usually are used in specific areas and these fields are not changing too much in 2006. As a result, the corresponding operating systems remain similar for their grassroots support. Some are

climbing up, such as Linux, Windows. These are "bright" points for those operating systems that are very good in PC environment and thus gain more support. Some curve is falling down, such as OS/2. It may because this is comparatively old operating system; with more and more modern systems emerged the old ones are going to loss their old support.

A very important fact is that the parameter matrices are not unique. All of these parameter matrices could be used to describe past and current trends of operating systems, because that is how they are constructed. But by replacing the data of 1997 by those of 2000 and replacing the data of 2000 by those of 2003, the results of 2006 are not the same by using different parameter matrices. That means each predictive derivation model will generate different prediction results. Although the predictions are not unique, these derivative models are still very helpful for researchers to understand the trends. After collecting more data in the future, these derivative models can be refined.

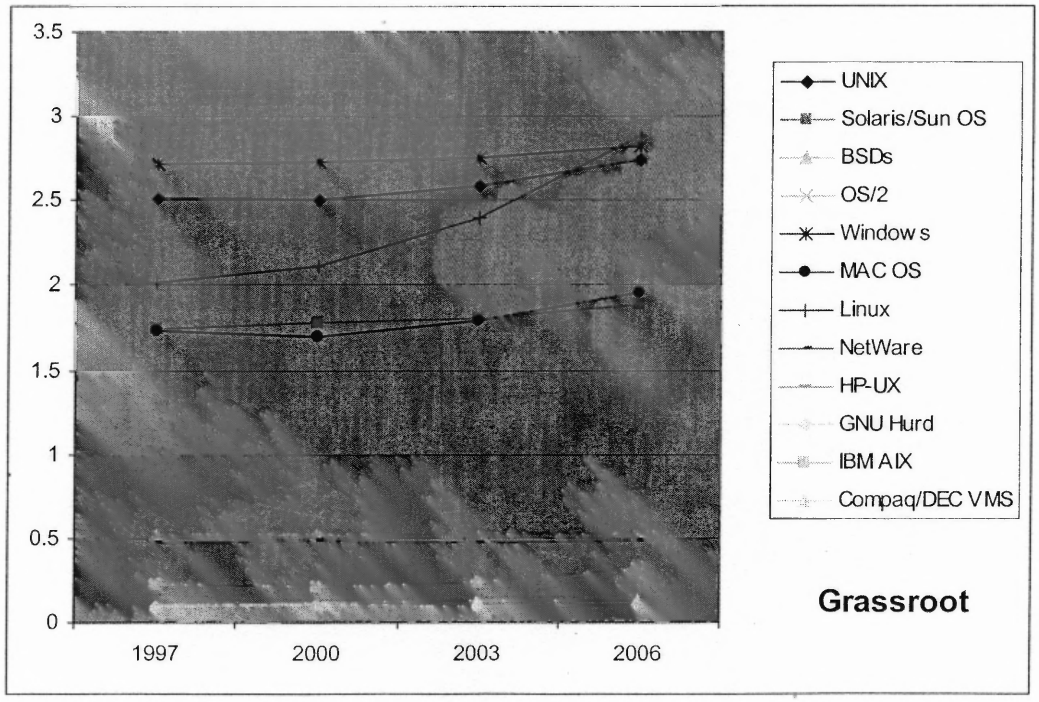


Figure 8.7 Trends of Grassroots Support from 1997 to 2006.

CHAPTER 9

EXTENDED FEATURE ANALYSIS

After the trends of individual operating system are predicted by considering all the independent variables, another perspective observation is worth looking at. What will the operating system looks like in the future? Or what features are important in the future? A more concrete example of this is, for instance, how does *openness* affect the evolution of operating systems? Or more specifically, how does *openness* affect the future organizational support of an operating system?

9.1 Extended Feature

In order to find out answers for such kinds of questions and provide some quantitative proof, the concepts of intrinsic factors and extrinsic factors need to be expanded. And thus comes the concept of Extended Feature. The extended feature is calculated in the following way:

Given an intrinsic factor and an extrinsic factor, the corresponding Extended Feature is defined as the sum of the product of an operating system's extrinsic factor and its intrinsic factor, including all operating systems under consideration.

So the mathematical representation is as follows:

$$Extended(feature, year) = \sum_{os \in OS} Extrinsic(os, year) * Intrinsic(feature, os)$$

Where:

Extended(feature, year): the extended feature of the year

feature: an intrinsic factor

os: an operating system in the candidate operating systems

Extrinsic(os, year): one extrinsic factor of the OS in the year

Intrinsic(factor, os): the intrinsic factor of the corresponding operating system

According to the above description, the properties of extended feature could be summarized as below:

- Extended Feature is related to one extrinsic factor and one intrinsic factor.
- Extended Feature changes over time.
- Extended Feature is not attached to one particular operating system. It is an overall concept that reflects characteristics of all operating systems in general.

Since most of the intrinsic factors are quantified in different dimensions, first they need to be normalized into a single magnitude. After normalization, the relationship between a certain intrinsic factor and extrinsic factor are considered. For a given extrinsic factor, a diagram for every intrinsic factor's variation through the years is picked out. By integrating all the curves for the same extrinsic factor in one diagram, the quantitative impact of different features on one extrinsic factor can be compared.

9.2 Analysis of Results

In Figure 9.1, some extended feature analysis for one extrinsic factor – organizational support are shown. In this group of analysis, five intrinsic factors are picked out: openness, security/protection, design, range of services and compatibility. For each of the intrinsic factor, the extended feature against organizational support for every observed year is derived. So that from the figure, not only the changes of extended features could be exposed, but also the comparison between extended feature of two different intrinsic factors could be discovered.

For example, in Figure 9.1, most of the lines are increasing through the year, which means that most of these intrinsic factors are important to extrinsic factor of organizational support and have more and more impact on organizational support. Or in other words, in order to attract the attention from organizations supports, one operating system need to enhance its feature of the above intrinsic factors. Recalling the method to judge organizational support, there are five different aspects that are been considered, for instance, organization standards, paper/articles published on/using the OS. Each of these sub-features actually is related to one or more of the above mentioned intrinsic factor, e.g. openness, design and range of system services vs. organization standards, security/protection and compatibility vs. paper/articles published on the OS, range of system services vs. paper/articles published using the OS, etc. Compare to other intrinsic features, the curve for range of services is not so steep. Probably this is because that range of system services that are provided by operating systems are getting more and more stable through the year and thus less paper/article are published on the

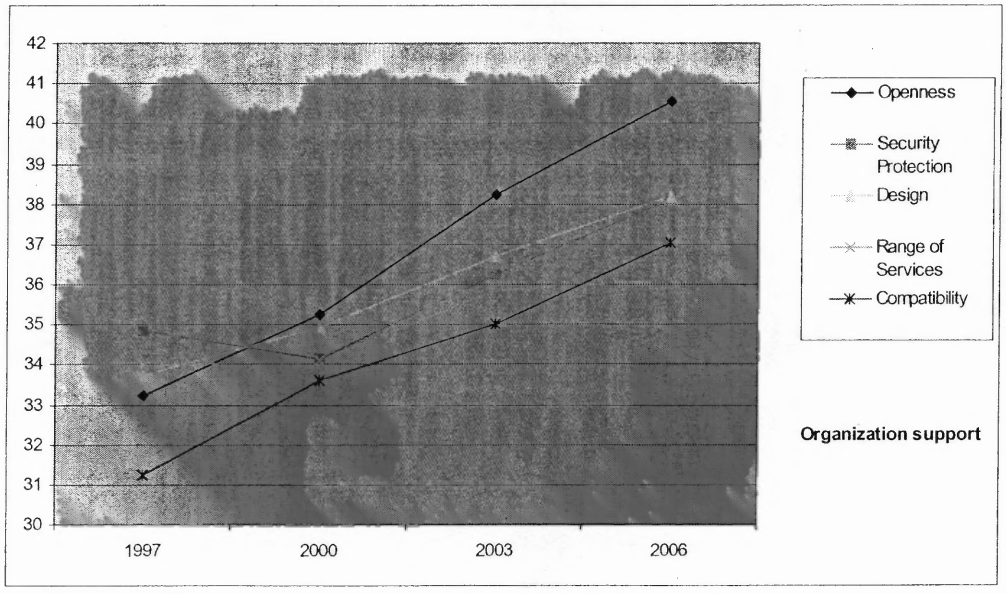


Figure 9.1 Sample Extended Feature Analysis for Organizational Support.

corresponding fields and not a lot of new standards are involving these topics.

Using the constructed statistical model that is discussed in previous chapters, the values of extended features for 2006 are derived. Range of services will remain similar; at the same time, most of other factors are getting more, with openness the top one.

From another perspective, different factors need to be compared horizontally. For example, in year 1997, *Extended*(Security/Protection) is greater than other values. However, in year 2000, the ranking for this value falls down to No.3 that almost as same as *Extended*(Range of Services), with *Extended*(openness) the top one, slightly bigger than the second one of *Extended*(Design). Different curves sometimes across each other through different years. This indicates that within different year, there could be different extended feature value for a single factor and a single support, as a result that within different year, the importance of a certain intrinsic factor against an extrinsic factor vary.

Another example is shown in Figure 9.2. Grassroots support is used and five

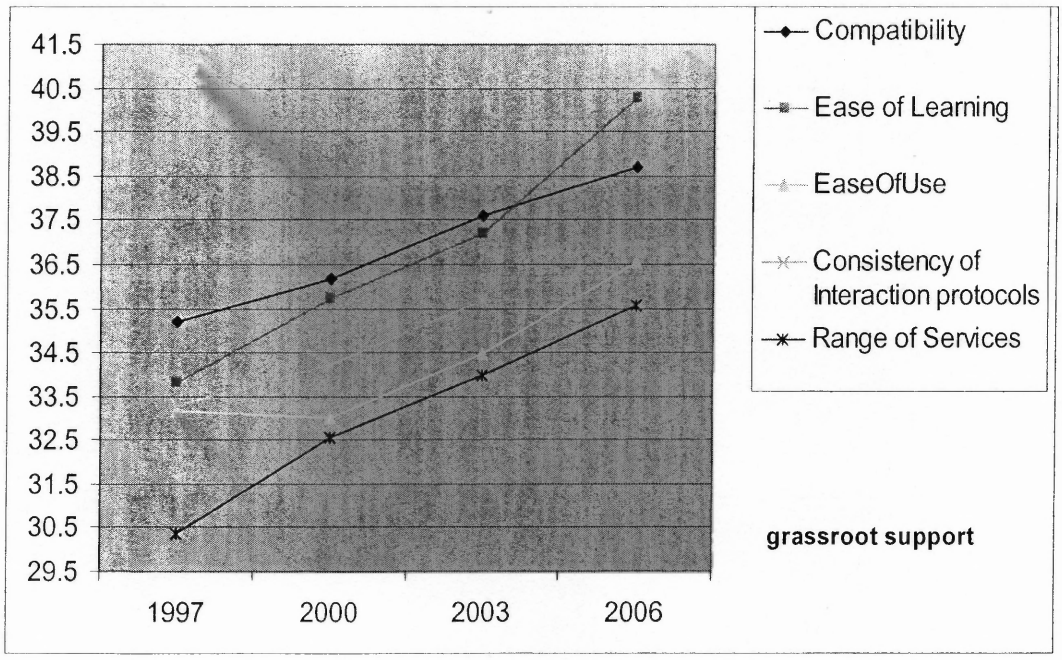


Figure 9.2 Sample Extended Feature Analysis for Grassroots Support.

intrinsic factors are selected as compatibility, ease of use, ease of learning, consistency of interaction protocol and range of services.

During the studied time periods, most of the values for the lines are increasing, which indicates the importance for these five intrinsic factors to grassroots support. Back to the definition of grassroots support, it includes the view from general computer users such as whether they know them, whether they use them or whether they like them. Therefore, the above factors are getting important. For example, people tend to like using those systems that are easy to learn and ease to use. Also, the wide the range of services and consistency of interaction protocols that an OS could provide the more convenient for people to use it and thus choose it.

It is predicted that in year 2006, *Extended*(ease of learning) will exceed compatibility and become the first one. *Extended* (Ease of use) will also raise and become more important. Consistency of interaction protocols remains similar.

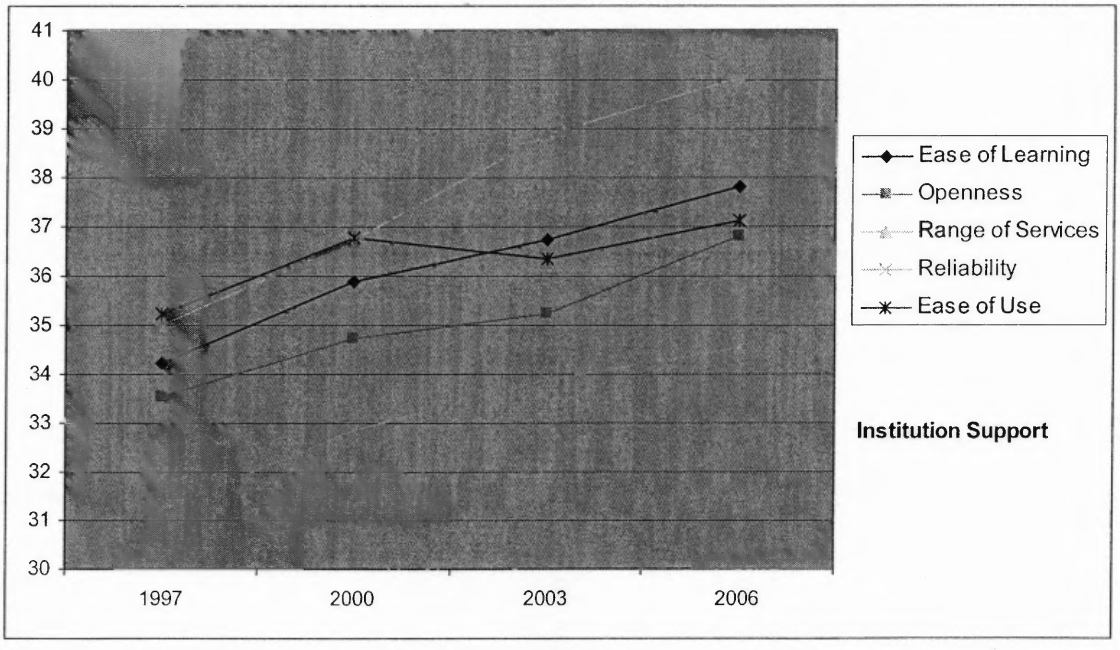


Figure 9.3 Sample Extended Feature Analysis for Institutional Support.

In Figure 9.3 ~ Figure 9.5, the samples of extended feature analysis for institutional support, industrial support and governmental support are presented respectively.

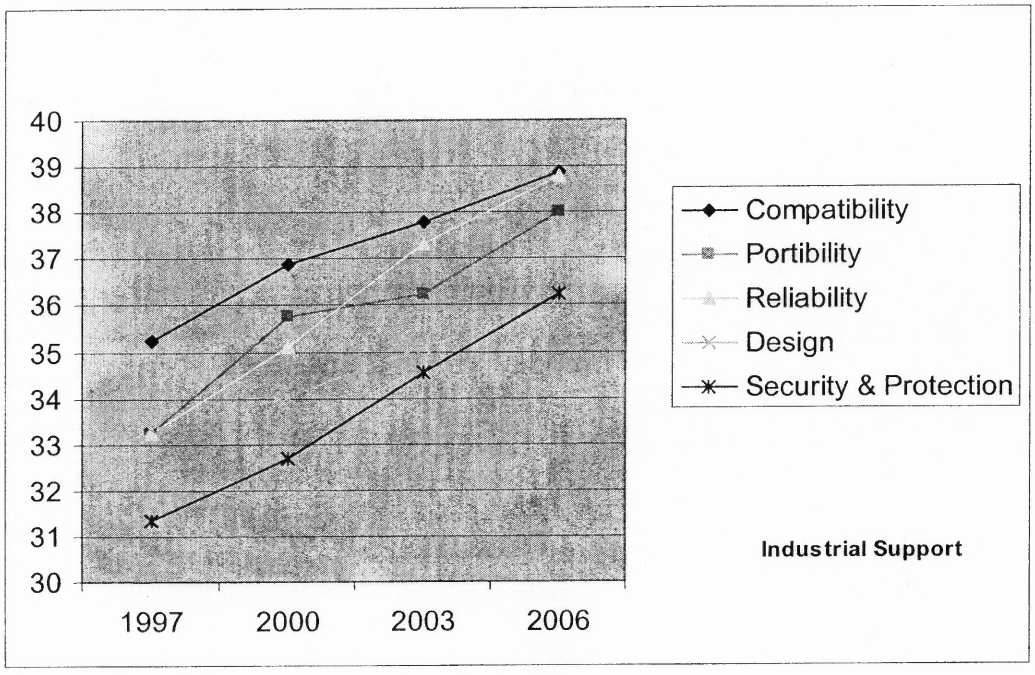


Figure 9.4 Sample Extended Feature Analysis for Industrial Support.

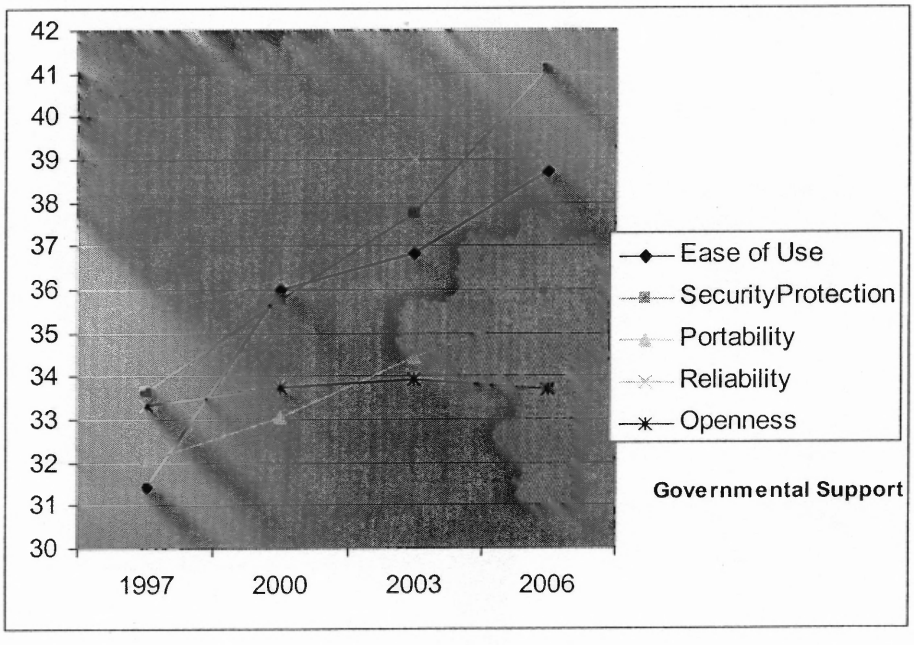


Figure 9.5 Sample Extended Feature Analysis for Governmental Support.

9.3 Conclusion

In this chapter, the following questions are addressed by introducing concept of Extended Feature: What will the operating system looks like in the future? What features are important in the future? Extended Feature is related to one extrinsic factor and one intrinsic factor and changes over time. Furthermore, Extended Feature is not attached to one particular operating system. It is an over-all concept that reflects characteristics of all operating systems in general.

By analyzing the extended features, the relationships between selected intrinsic factors and the given extrinsic factor can be concluded as following:

- *Openness*, *security/protection*, *design*, *range of services* and *compatibility* are important features for organizational support and will continue to be crucial for organizational support in 2006. Among these factors, *openness*, *security/protection* and *design* will increasingly be more important than *range of services* and *compatibility* in 2006.

- *Ease of use*, *ease of learning*, *compatibility*, *consistency of interaction protocols* and *range of services* are important features for grassroots support and will continue to be crucial for grassroots support in 2006. Among these factors, *ease of use*, *ease of learning* and *compatibility* will increasingly be more important than *range of services* and *consistency of interaction protocols* in 2006.

- *Ease of use*, *ease of learning*, *reliability*, *openness* and *range of services* are important features for institutional support and will continue to be crucial for institutional support in 2006. Among these factors, *range of services*, *ease of learning*, *ease of use*, and *openness* will increasingly be more important than *reliability* in 2006.

- *Compatibility, reliability, portability, design* and *security/protection* are important features for industrial support and will continue to be crucial for industrial support in 2006. Among these factors, *Compatibility, reliability* and *portability* will increasingly be more important than *design* and *security/protection* in 2006.

- *Security/protection, reliability, ease of use, portability* and *openness* are important features for governmental support and will continue to be crucial for governmental support in 2006. Among these factors, *Security/protection, reliability* and *ease of use* will increasingly be more important than *portability* and *openness* in 2006.

CHAPTER 10

MODEL VALIDATION AND IMPROVEMENTS

10.1 Model Validation

In this empirical study, operating systems trends were considered, their evolution over time was observed, the evolution by means of time series was recorded, and the general statistics models for how these trends evolve and feature analysis have been constructed.

After the statistics models for operating system evolution have been constructed, a number of methods need to be introduced to assess the reliability of these models. Assessing the quality of a model is called model validation. Model validation is something that needs to be done both by producers and users of models. A model is just a human being's hypothesis of a simplified representation of the real world, so it is always a good practice to do validation to check the reliability of a model.

In this project, the following methods will be used to validate the statistics models:

- Check the difference between the actual values and the predictive values from the statistical models.
- Use historical data of the other operating systems that are not in the operating system list to validate and correct the model.
- Revise these models based on the newest evolution of operating system.

F-Statistic[134,135], which is a standard statistical method to check if there are significant differences between groups, is used to validate the prediction.

In the F-table, for significant level $\alpha = 0.05$, F must be greater than 4.49 to reject the hypothesis of statistical correlation; because the value of F is much less, the hypothesis is validated.

Table 10.1 Difference between Actual Value and Predictive Value in 2003

OS	Government Support	
	Actual Data	Predicted Data
UNIX	2.575	2.532
Solaris	2.061	2.013
BSDs	1.561	1.449
OS/2	0.515	0.483
Windows	2.655	2.674
MAC OS	0.343	0.095
Linux	2.735	2.747
NetWare	1.052	1.020
HP-UX	2.262	2.204
GNU Hurd	0.492	0.432
IBM AIX	2.316	2.268
Compaq DEC VMS	1.493	1.426

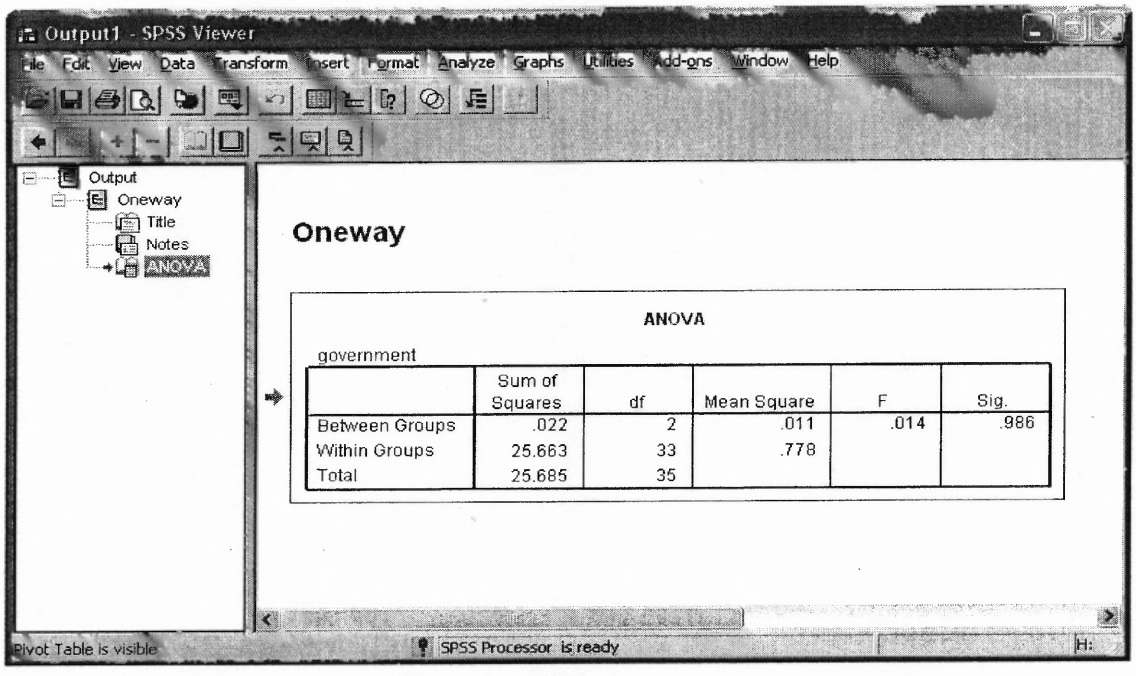


Figure 10.1 F-value Validation.

Table 10.1 shows the difference between the actual values and the predictive values for the extrinsic factor of Governmental Support. It shows that the predictive

values match actual values very well and this statistical model is valid to describe the historical and current trends of operating systems.

10.2 Model Improvement

10.2.1 Weakness

There are no perfect statistics models. A famous statistic statement is: "All models are wrong, but they are useful." Instead of perfect models, reasonable models are maintained to describe the historical trends and current evolution of operating systems. Somehow, these models can also be used as good references to predict the future trends of operating systems. This multiple regression model for operating systems trends has the following weaknesses.

The first weakness of the regression model is that the number of the variables is larger than the number of observations. It makes the regression models to be not unique. To avoid that problem, two ways will be used to refine the models. Most correlated variables are always manually selected into the model first. SPSS will also automatically delete the dependent 19 variables until variables (equal to the number of the observations) can be decided. In the future, it is hoped that more operating system will be included into the research so the number of observation can increase and make the model sounder.

The second weakness is that the data may not in proper format or not complete. In the data survey process, some problems are difficult to solve. And bias may be caused. For example, not every company responds to the survey, so the information is incomplete. If the silent companies are tending more to use one kind of specific operating system, a

bias will be caused by the available data. Also, it is hard to get cooperation from government, because they are not willing to participate in the survey.

The features of the operating systems may prove that multiple regression models are not the best model for analysis. The feature of a new operating system may appear to be like the description below: "An operating system will first expand after it's created, increasing even in exponent way. Then, it goes to the summit in a few years. As time is going on and newer operating system comes out, it begins a decreasing period. Then after a change period, it keeps somehow constant. There will not have much vibration for the rest of its life. The lifecycle of an operating system likes a bell curve."

This description is reasonable based on the common sense of an operating system. However, it is far from what is obtained in this multiple multivariate models. So that multiple multivariate regression may not be very appropriate here, or can be considered just as an approximation. However, more data is needed to track the whole process instead of three periods.

10.2.2 Possible Improvement

To investigate the future data, more observations are needed. And based on more data, TIME SERIES method can be used to construct better statistics models. There are two main goals of time series analysis: identifying the nature of the phenomenon represented by the sequence of observations, and predicting future values of the time series variable. Both of these goals require that the pattern of observed time series data is identified and more or less formally described.

Time series data often arise when monitoring industrial processes or tracking corporate business metrics. Time series analysis accounts for the fact that data points

taken over time may have an internal structure, such as autocorrelation, trend or seasonal variation. The below introduction of Time Series: [133,134,135]

Time series model is an ordered sequence of values of a variable at equally spaced time intervals. The usage of time series models is two fold:

- Obtain an understanding of the underlying forces and structure that produced the observed data.
- Fit a model and proceed to forecasting, monitoring or even feedback and feed forward control.

The fitting of time series models can be an ambitious undertaking. There are many kinds of models, such as Box-Jenkins ARIMA models, Box-Jenkins Multivariate Models, Holt-Winters Exponential Smoothing (single, double, triple), and Box-Jenkins ARIMA models.

The term "univariate time series" refers to a time series that consists of single (scalar) observations recorded sequentially over equal time increments. Some examples are monthly CO_2 concentrations and southern oscillations to predict El Nino effects.

Although a univariate time series data set is usually given as a single column of numbers, time is in fact an implicit variable in the time series. If the data are equi-spaced, the time variable, or index, does not need to be explicitly given. The time variable may sometimes be explicitly used for plotting the series. However, it is not used in the time series model itself.

Box-Jenkins Multivariate Models

The multivariate form of the Box-Jenkins univariate models is sometimes called the ARMAV model, for Auto Regressive Moving Average Vector or simply vector ARMA process.

The ARMAV model for a stationary multivariate time series, with a zero mean vector, represented by

$$x_t = (x_{1t}, x_{2t}, ..., x_{nt})^T \qquad -\infty < t < \infty$$

is of the form

$$\mathbf{x}_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p} + a_t$$
$$- \theta_1 a_{t-1} - \theta_2 a_{t-2} - \ldots - \theta_q a_{t-q}$$

Where

$$\phi_k = \{\phi_{k.jj}\}, \qquad k = 1, 2, \ldots, p$$

$$\theta_k = \{\theta_{k.jj}\}, \qquad k = 1, 2, \ldots, q$$

x_t and at are n x 1 column vectors with at representing multivariate white noise are n x n matrices for autoregressive and moving average parameters

E[at] = 0

$$E(a_t a'_{t-k}) = 0 \qquad k \neq 0$$

$$E(a_t a'_{t-k}) = \Sigma_a \qquad k = 0$$

where a is the dispersion or covariance matrix of at a_t

As an example, for a bivariate series with n = 2, p = 2, and q = 1, the ARMAV(2,1) model is:

$$\begin{pmatrix} x_{1t} \\ x_{2t} \end{pmatrix} = \begin{pmatrix} \phi_{1.11} & \phi_{1.12} \\ \phi_{1.21} & \phi_{1.22} \end{pmatrix} \begin{pmatrix} x_{1t-1} \\ x_{2t-1} \end{pmatrix} + \begin{pmatrix} \phi_{2.11} & \phi_{2.12} \\ \phi_{2.21} & \phi_{2.22} \end{pmatrix} \begin{pmatrix} x_{1t-2} \\ x_{2t-2} \end{pmatrix} + \begin{pmatrix} a_{1t} \\ a_{2t} \end{pmatrix} - \begin{pmatrix} \phi_{1.11} & \phi_{1.12} \\ \phi_{1.21} & \phi_{1.22} \end{pmatrix} \begin{pmatrix} a_{1t-1} \\ a_{2t-1} \end{pmatrix}$$

With

$$a_t = \begin{pmatrix} a_{1t} \\ a_{2t} \end{pmatrix}$$

The estimation of the matrix parameters and covariance matrix is complicated and very difficult without computer software. The estimation of the Moving Average matrices is especially an ordeal. The ARV model is given without the MA components:

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p} + a_t$$

Where

$$\phi_k = \{\phi_{k.jj}\}, \qquad k = 1, 2, \ldots, p$$

x_t is a vector of observations, x_{1t}, x_{2t}, ... , x_{nt} at time t

a_t is a vector of white noise, a_{1t}, a_{2t}, ... , a_{nt} at time t

is a n x n matrix of autoregressive parameters

$E[a_t] = 0$

$$E(a_t a'_{t-k}) = 0 \qquad k \neq 0$$

$$E(a_t a'_{t-k}) = \Sigma_a \qquad k = 0$$

Where $\sum_a = E[a_t, a_{t-k}]$ is the dispersion or covariance matrix

A model with p autoregressive matrix parameters is an ARV(p) model or a vector AR model. The parameter matrices may be estimated by multivariate least squares, but there are other methods such as maximum likelihood estimation.

There are a few interesting properties associated with the phi or AR parameter matrices. Consider the following example for a bivariate series with n =2, p = 2, and q = 0. The ARMAV(2,0) model is:

$$\begin{pmatrix} x_t \\ y_t \end{pmatrix} = \begin{pmatrix} \phi_{1.11} & \phi_{1.12} \\ \phi_{1.21} & \phi_{1.22} \end{pmatrix} \begin{pmatrix} x_{t-1} \\ y_{t-1} \end{pmatrix} + \begin{pmatrix} \phi_{2.11} & \phi_{2.12} \\ \phi_{2.21} & \phi_{2.22} \end{pmatrix} \begin{pmatrix} x_{t-2} \\ y_{t-2} \end{pmatrix} + \begin{pmatrix} a_{1t} \\ a_{2t} \end{pmatrix}$$

Without loss of generality, assume that the X series is input and the Y series are output and that the mean vector = (0,0). Therefore, transform the observation by subtracting their respective averages.

The diagonal terms of each Phi matrix are the scalar estimates for each series, in this case:

$\phi_{1.11}$, $\phi_{2.11}$ for the input series X, $\phi_{1.22}$, $\phi_{2.22}$ for the output series Y.

The lower off-diagonal elements represent the influence of the input on the output. This is called the transfer mechanism or transfer-function model as discussed by Box and Jenkins. The terms here correspond to their terms.

The upper off-diagonal terms represent the influence of the output on the input. This is called "feedback". The presence of feedback can also be seen as a high value for a coefficient in the correlation matrix of the residuals. A "true" transfer model exists when there is no feedback. This can be seen by expressing the matrix form into scalar form:

$$x_t = \phi_{1.11} x_{t-1} + \phi_{2.11} x_{t-2} + \phi_{1.12} y_{t-1} + \phi_{2.12} y_{t-2} + a_{1t}$$

$$y_t = \phi_{1.22} u_{t-1} + \phi_{2.22} y_{t-2} + \phi_{1.21} x_{t-1} + \phi_{2.21} x_{t-2} + a_{2t}$$

Finally, delay or "dead" time can be measured by studying the lower off-diagonal elements again. If, for example, $\phi_{1.21}$ is non-significant, the delay is 1 time period.

Holt-Winters Exponential Smoothing (single, double, triple)

This is a very popular scheme to produce a smoothed Time Series. Whereas in Single Moving Averages the past observations are weighted equally, Exponential Smoothing assigns exponentially decreasing weights as the observation get older. In other words, recent observations are given relatively more weight in forecasting than the older observations.

In the case of moving averages, the weights assigned to the observations are the same and are equal to 1/N. In exponential smoothing, however, there are one or more smoothing parameters to be determined (or estimated) and these choices determine the weights assigned to the observations.

By previous analysis, Time Series method can focus on the internal trend of the data, which is just the purpose – to find the internal trend of the development of an operating system. However, Time Series requires much more data. It needs a long time following and correct recording.

CHAPTER 11

CONCLUSION AND FUTURE WORK

11.1 Summary

In this dissertation, a tentative effort has been discussed to characterize operating system evolution and how they evolve.

In Chapter 2, the following questions are point out:

- Is it possible to predict if an operating system will succeed or fail?
- What will be the operating systems that will be studied?
- What are the possible factors which can affect the trend?
- What information should be collected in order to determine if an operating system succeed or fail?
- How to quantify the factors and find a model/function to predict the trends?
- What are the results of the evolution analysis for operating system?
- Beyond the analysis of the evolution of individual operating system, the evolution of operating system features is also analyzed. So that, even if operating system's success or failure can not be told, the future of operating systems can be characterized by their main attributes.

From this project, the answers for the above questions are addressed:

- Yes, it is possible to predict if an operating system will succeed or fail.
- 15 selected operating systems are investigated. They are UNIX, Solaris/Sun OS, BSDs (including FreeBSD, OpenBSD, NetBSD), OS/2, Windows, MS-DOS, MAC OS, Linux, NetWare, HP-UX, GNU Hurd, IBM AIX, Compaq/DEC VMS, Multics, OS360.
- Information of two categories of factors: Intrinsic factors and extrinsic factors are collected to predict the trends of operating systems.
- Five different quantifying methods are applied on all the factors and a statistical model is built to predict the trends.

- The detailed results of the evolution analysis for operating system are discussed in Chapter 7 and Chapter 8.

- Beyond the analysis of the evolution of an individual operating system, evolution of operating system features are analyzed by introducing the concept of extended features. Thus the future of operating systems is characterized by their main attributes.

As part of computing engineering technology evolutions, this dissertation is concentrated on a family of the evolutions: operating systems. First, the author discussed what could be the possible factors that can affect the trends of operating systems. The evolution of operating systems is affected by a dizzying array of factors, which are themselves driven by a wide range of sources, such as market forces, corporations, government agencies, standards bodies, academics, etc. In author's point of view, both intrinsic factors and extrinsic factors would have impact on the evolution of operating systems. After discussing the definition of intrinsic factors and extrinsic factors, a group of factors that could be used to watch operating systems trends are identified.

In order to use empirical method to analyze the evolution, an approach must be found to quantify intrinsic factors and extrinsic factors of operating systems. To quantify intrinsic factors, all features of an operating system are reviewed to check if they match these factors, and scores are assigned for them. Extrinsic factors are not the same as intrinsic factors. Basically, they are questions for different fields. Extrinsic factors are questions that ask for the numbers, so the answers will be used as the value of this extrinsic factor.

Different quantifying methods have been applied to different factors according to their nature. For intrinsic factors, the methodologies include numeric formula,

hierarchical sub-feature, cumulative sub-feature and discrete scale sub-feature. For extrinsic factor, similar techniques are applied.

Furthermore, collecting data involves many resources and approaches. Text books, authorial books; system manuals and hand books; journals, papers and other articles; internet resources all have been referenced. In addition, for those that are lack of resources or do not match the requirements for this project, a survey web has been set up for the purpose.

A set of operating systems (15) has been selected, and intrinsic factors are evaluated based on the applicable version of each operating system. For extrinsic factors, surveys have been done for each field of them in 1997, 2000 and 2003. The value of intrinsic factor will not change during the time, while the value of extrinsic factor does change in different period. All data that are collected through survey webpage are stored in the database. Check the complete survey results at: http://swlab.njit.edu/OS/survey.htm.

Based on all the collected data, statistics methods are used to analyze these data. *Principle components analysis* (PCA) models and *Canonical Correlation analysis* are constructed to analyze the data and describe the relationships among these factors and the historical advancements of each operating system. Correlation among these factors has been analyzed and new independent components are constructed by using factor analysis. Multiple regression method has been used to construct the statistics models for operating system evolutions.

Beyond the analysis of the evolution of individual operating system, the evolution of operating system features are also analyzed. So that, even if an operating system's

success or failure can not be told, the future of operating systems can be characterized by their main attributes.

After statistics models that are constructed will be extended to do provisional prediction for future trends and characterizing the future attributes. The prediction model can be validated by future data.

11.2 Evaluation

In this study for operating system evolution characterization, the author merely attempt to capture observed behaviors by empirical laws. After having collected enough data and constructed statistics models, the trends of operating systems and the individuality of operating system attributes could be understood better. By factor analysis, each factor gets a parameter. The guess is that the factors with larger parameter will have bigger impact to evolution of operating systems. By this means, the following conclusions can be drawn:

- From the statistics models, generally, the parameters of extrinsic factors are greater than the parameters of intrinsic factors. So, extrinsic factors have bigger impact than intrinsic factors.

- Extrinsic factors can be used to check if an operating system succeeds or fails. They are also very important factors to predict the future evolution of an operating system. In author's point of view, if an operating system can earn support from the majority of one field, it can be called a successful operating system, such as Sun Solaris, UNIX. If an operating system can earn support from the majority of grassroots and every field, it can be considered as very successful operating system, such as Windows, Linux.

- Although intrinsic factors have less impact than extrinsic factors, they do impact the trends of operating systems, especially security & protection, openness, and ease of use.

Besides, the concept of extended feature is defined and analyzed and compare different extended features are compared. How the intrinsic factors have affected the evolution of extrinsic factors is found out.

The following conclusions could be drawn for extended feature analysis:

- Extended feature can be used to analyze the relationship between intrinsic factors and certain extrinsic factors.
- Because extended feature is not bound to a particular operating system, it can be utilized to analyze intrinsic factors in general rather than one specific to an operating system.
- The relative value of different extended feature can be used to rank the importance of different extended feature as to the evolution of an operating system.

Another very important factor is the time. The data collected show that the feature of a new operating system may appear to be like the description below: When an operating system is introduced, there are some enthusiasts who are willing to learn it. The operating system could expand at first, increases even in exponent ways. After a while, there will be more people who would like to use this operating system. Then, as newer operating systems come out, the older operating system will pass its pinnacle and begin to decrease. After a changing period, it keeps somewhat constant because there are certain groups of people who would like to stick to this operating system. The operating system will exist without much vibration in the rest of its life. The life cycle of an operating system is very similar with Geoffrey A. Moore's technology adoption life cycle[137]. Figure 11.1 shows the general lifecycle of an operating system.

The bell curve in Figure 11.1 is very useful as the general analytical model for the trends of operating systems. The empirical results could be used to explain why a

successful operating system could earn the support from majority programmer and why the other ones failed although a lot of innovators support them.

11.3 Future Work

A lot of data and interesting survey results have been collected in this empirical research for operating system trends, but they are still not enough. If more data could be collected in the future, they can be used to improve current statistics models. Instead of using correlation and regression models, more advanced statistics methods, such as time series method, can be used to improve current models in the future.

Empirical study for operating systems is just an exploratory beginning of the whole project of computing engineering trends. After using empirical method, analytical method will be used for operating system trends. Future work will not only attempt to capture observed behaviors by empirical laws, but also attempts to understand the phenomena that underlie observed behavior and build models that capture these phenomena.

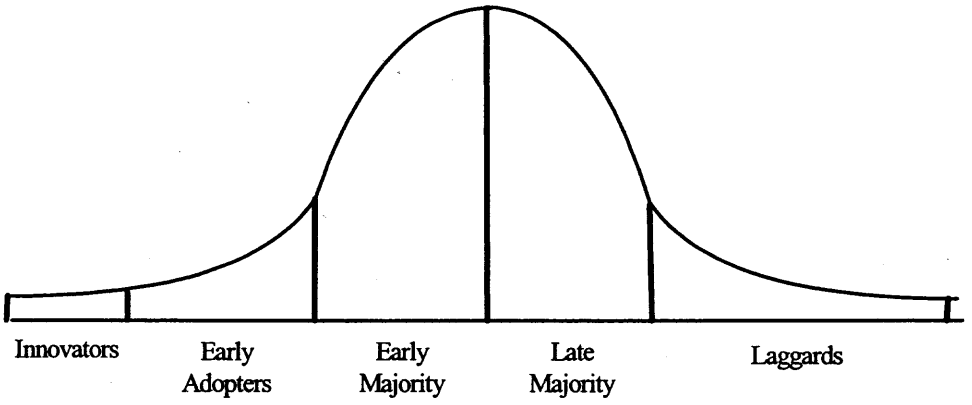


Figure 11.1 General Lifecycle for Technique Adoption.

After studying the trends of operating systems, the other fields of software engineering will be done in near future. For example: the trends of networking, the trends of database, the trends of management system, etc. All of these trends will use the similar methods to analyze. Generally, empirical method will be used first. After having better understanding of trends behavior, analytical method will used for them to understand the cause/effect relationships.

REFERENCES

[1] Y.Chen, A.Mili, L.Wu, R.Dios, K.Wang. Programming Language Trends: an Empirical Study. Submitted to 26th International Conference on Software Engineering.

[2] H.Lorin, H.M.Deitel. Operating Systems. Addison-Wesley Publishing Company, Inc, 1981.

[3] Gary J.Nutt. Operating Systems: A Modern Perspective. 2nd ed. 2000.

[4] Andrew S.Tanenbaum. Modern Operating System. Second Edition ed. Upper Saddle River, New Jersey: Prentice Hall, 2001.

[5] Avi Silberschatz, Peter Galvin, Greg Gagne. Operating System Concepts. Sixth ed. John Wiley & Sons, Inc, 2003.

[6] Carnegie Mellon Software Institute. View the Quality Measures Taxonomy. http://www.sei.cmu.edu/str/indexes/glossary. 2003.

[7] Intel Corporation. Intel Solution Services white paper: Linux Scalability. http://www.intel.com/internetservices/intelsolutionservices/downloads/linux_scalability.pdf. 2003.

[8] Donald R.Horner. Operating Systems: Concepts and Applications. London: Scott, Foresman and Company, 1989.

[9] Stanley A.Kurzban, Thomas S.Heines, Anthony P.Sayers. Operating Systems Principles. Second ed. New York: Van Nostrand Reinhold Company, 1984.

[10] Andrew S.Tanenbaum, Albert S.Woodhull. Operating Systems, Design and Implementation (second edition). Prentice Hall, 1996.

[11] Christian Green. "EASE OF USE", http://product.info.apple.com/productinfo/macadvantage/50advantages/ease.html. 1996.

[12] S.Cranefield MPMNaPH. Ontologies for interaction protocols. In Proceedings of the Workshop on Ontologies in Agent Systems 2002.

[13] The Open Group. Operating System Services; http://www.opengroup.org. 1998.

[14] Microsoft MSDN, http://msdn.microsoft.com. 2003.

[15] The OS/2 WWW Homepage, http://www.mit.edu:8001/activities/os2/os2world.html. 2004.

[16] OS/2 Wrap, http://www-3.ibm.com/software/os/warp/. 2003.

[17] Frederick P.Brooks J. The Mythical Man-Month. Addison Wesley Longman, Inc.,1995.

[18] IBM. Design Basic, http://www-306.ibm.com/ibm/easy/eou_ext.nsf/Publish/6. 2004.

[19] Kingsley Martin. Total Asset Administration, http://www.ljx.com/ltpn/october97/total_p32.html. 1998.

[20] John Kirsch. Microsoft Windows NT Server 4.0 versus UNIX. http://www.unix-vs-nt.org/. 1998.

[21] Lubomir Bic, Alan C.Shaw. The Logical Design of Operating Systems. 2nd ed. Prentice Hall, 1988.

[22] Olczak, Anatole. Unix and Linux programming manual. Addison-Wesley, 2001.

[23] Olczak A. The Korn Shell: Unix and Linux programming manual /. Addison-Wesley, 2001.

[24] UNIX system V/386, release 3.2: programmer's reference manual. 1988. Prentice Hall.

[25] Stephen Whalley. Making PCs easier to use. ftp://download.intel.com/technology/easeofuse/eous2ps.pdf. Intel Corporation, 1999.

[26] User Interface Consistency Checklist; http://www.csc.calpoly.edu/~jdalbey/SWE/QA/UIConsistency.html. 2002.

[27] Yannis Grammatis. Network Hierarchy. http://www.chaminade.org/MIS/lanhier.htm. 1998.

[28] The XFree86 Project http://www.xfree86.org/. 2004.

[29] The UNIX System http://www.unix-systems.org/. 2004.

[30] The ISBN Standards http://www.isbn.org/standards/home/index.asp. 2004.

[31] IEEE Standards Association http://standards.ieee.org/. 2004.

[32] IEEE Online; http://www.ieee.org. 2004.

[33] Free Soft, http://www.freesoft.org/. 2004.

[34] Association of Computing Machinery Digital Library; http://www.acm.org/dl. 2004.

[35] The Open Group http://www.opengroup.org/. 2004.

[36] Gancarz M. Linux and the Unix philosophy. Amsterdam; Boston: Digital Press, 2003.

[37] Ken Frazer. History of Solaris http://home.earthlink.net/~krfrazer3/Solaris_History_2per.pdf. 2003.

[38] The BSD Family Tree; http://www.daemonnews.org/200104/bsd_family.html. 2003.

[39] Free BSD Official Homepage. http://www.freebsd.org/, http://www.freebsd.com/. 2004.

[40] FreeBSD Forums http://www.freebsdforums.org. 2004.

[41] NetBSD Manual Pages, http://netbsd.gw.com/cgi-bin/man-cgi?++NetBSD-current. 2001.

[42] NetBSD website http://www.netbsd.org/. 2004.

[43] FreeBSD Handbook, http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/. 2003.

[44] The FreeBSD Handbook, http://www.freebsd.org/handbook/handbook.html. 2004.

[45] OpenBSD website http://www.openbsd.org/. 2004.

[46] Marshall Kirk McKusick, Keith Bostic, Michael J.Karels, John S.Quarterman. The design and implementation of the 4.4BSD operating system. Reading, Mass.: Addison-Wesley, 1996.

[47] Thierry Falissard. MVS... a long history: OS/360; http://mcraeclan.com/Links/Computers/IBMMainframeHistory/mvshist1.htm. 2003.

[48] Microsoft Corp. Architectural Services Firm Draws on .NET Platform to Retain Competitive Edge, Trim TCO, Build IT Functionality. 2002.

[49] Microsoft Corporation http://www.microsoft.com. 2004.

[50] A Short History of MS-DOS http://www.patersontech.com/Dos/Byte/History.html. 1988.

[51] Craig Danuloff MRB. The Mac OS 8 Book: The Ultimate Macintosh User's Guide. Ventana Communications Group Inc., 1997.

[52] Robin Williams. The Little Mac Book. 5th edition ed. Peachpit Press, 1998.

[53] Mark G.Sobell. A Practical Guide to Linux. Addison-Wesley Pub Co, 1997.

[54] John Purcell, Amanda Robinson. Linux Encyclopedia. 5th edition ed. Independent Pub Group (Computer); 1997.

[55] Steve Oualline. Discover Linux. IDG Books Worldwide, 1997.

[56] Petersen R. Linux: the complete reference. Berkeley, Calif. Osborne, 2001.

[57] History of Linux (version 2.1). 2003.

[58] The Linux Home Page, http://www.linux.org/. 2003.

[59] Linux Online http://www.linux.org. 2004.

[60] James E.Gaskin JGJG. Mastering Netware 5.1. Sybex, 2000.

[61] Onword Press Development Team JR. Five Steps to HP-UX. OnWord Press, 1993.

[62] Martin Poniatowski. Learning the HP-UX Operating System. Hall Press, 1996.

[63] HURD home http://www.gnu.org/home.html. 2003.

[64] The GNU Hurd - GNU Project - Free Software Foundation (FSF), http://www.gnu.org/software/hurd. 2003.

[65] IBM AIX: UNIX operating system - an open UNIX solution, http://www.ibm.com/servers/aix/. 2003.

[66] IBM AIX 5L. http://www-1.ibm.com/servers/aix/. 2004.

[67] Andreas Siegert. The AIX Survival Guide. Addison-Wesley Pub Co, 11001.

[68] David Donald Miller. Open VMS Operating System Concepts. 2nd edition ed. Digital Press, 1997.

[69] OpenVMS homepage. http://www.openvms.digital.com/. 2004.

[70] http://www.OpenVMS.org. 2004.

[71] Multics Home page, http://www.mit.edu:8001/afs/net/user/srz/www/multics.html. 2000.

[72] Multics, http://www.multicians.org/ . 2003.

[73] OS/2 Supersite; http://www.os2ss.com. 2001.

[74] George Eckel. Inside Windows NT workstation. Indianapolis, Ind.: New Riders Pub., 1996.

[75] Windows XP and Windows .NET Server Technical Overview. http://www.studentconsultant.org/germany/augsburg/files/Win-TechnicalOverview.pdf. 2003.

[76] HP-UX 11i Operating System, http://www.hp.com/products1/unix/operating/. 2003.

[77] Robbins A. UNIX in a nutshell: a desktop quick reference for System V Release 4 and Solaris 7. O'Reilly, 1999.

[78] Uresh Vahalia. UNIX Internals: the new frontiers. Prentice-Hall, Inc, 1996.

[79] Unix Manual. 2000.

[80] Ray Duncan. The MS-DOS encyclopedia. Redmond, Wash.: Microsoft Press, 1988.

[81] Watters PA. Solaris 8: the complete reference. McGraw-Hill, 2000.

[82] Paul A.Watters. Solaris 9: The Complete Reference (1st edition). McGraw-Hill Osborne Media, 2002.

[83] Mauro J. Solaris Internals: core kernel components. Palo Alto, CA: Sun Microsystems, Inc., 2001.

[84] NOVELL: Novell NetWare 6.5, http://www.novell.com/products/netware/. 2003.

[85] Linux Online: Solution and Sizing. http://www.linux.org/docs/ldp/howto/HP-HOWTO/sizing.html. 2003.

[86] Moody G. The rebel code: the inside story of Linux and the open source revolution. Cambridge, Mass.: Perseus Pub., 2001.

[87] McMahan S. Automating Windows with Perl. Lawrence, Kan. R&D Books ; Emeryville, CA : Distributed in the U.S. and Canada by Publishers Group West, 1999.

[88] Wyke RA. The Perl 5 programmer's reference: Windows 95/NT, Macintosh, OS/2 & UNIX. Research Triangle Park, NC: Ventana Communications Group, 1997.

[89] Ledgard HF. ADA, a first introduction. New York: Springer-Verlag, 1986.

[90] Kearney DS. The ADA in practice. Kingston, MA: R.S. Means Co., 1995.

[91] Chapman RB. OS/2 presentation manager programming for COBOL programmers. Boston: QED Pub. Group, 1993.

[92] Stroustrup B. The C++ programming language. Reading, Mass: Addison-Wesley, 1995.

[93] Kernighan BW. The C programming language. Englewood Cliffs, N.J.: Prentice Hall, 1988.

[94] Andrews M. C++ Windows NT programming. New York, N.Y.: M&T Books, 1996.

[95] Lindsey CH. Informal introduction to ALGOL 68. New York: North-Holland Pub. Co., 1977.

[96] Pagan FG. A practical guide to Algol 68. London; New York: Wiley: Wiley, 1976.

[97] Arnold K. The JAVA programming language. Reading, MA: Addison-Wesley, 1998.

[98] Meyers N. Java programming on Linux. Waite Group Press, 2000.

[99] Apple - Mac OS X, http://www.apple.com/macosx/. 2003.

[100] GNU Hurd 0.2 Kernel Source Tour, http://tamacom.com/tour/hurd/. 2003.

[101] Mac OS X Development, http://developer.apple.com/macosx/. 2003.

[102] Hewlett-Packard Technical Documentation, http://docs.hp.com/. 2003.

[103] Jim Boyce. Microsoft Windows 98 user manual. Indianapolis, IN, 1998.

[104] Rob Tidrow. Windows 98 installation and configuration handbook. Indianapolis, Ind, 1998.

[105] Tim O'Reilly, Troy Mott, Mott T. Windows 95 in a nutshell: a desktop quick reference. Cambridge, O'Reilly, 1998.

[106] Duane Hellums, Hellums Duane. Red Hat Linux: installation and configuration handbook. Indianapolis, Ind. Que Macmillan USA, 2000.

[107] Nazeeh Amin El-Dirghami, Youssef A.Abu Kwaik. SuSE Linux installation and configuration handbook. Indianapolis, Ind.: Que, 2000.

[108] Bob DuCharme. The operating systems handbook: UNIX, OpenVMS, OS/400, VM, and MVS. New York: McGraw-Hill, 1994.

[109] Greg Lehey. Porting UNIX software: from download to debug. Sebastopol, CA: O'Reilly & Associates, 1995.

[110] Dale Dougherty, Richard Koman. The Mosaic handbook for the Macintosh. Sebastopol, CA: O'Reilly & Associates, Inc., 1994.

[111] Kelley J.P.Lindberg, Jeffrey L.Harris. Novell's NetWare 6 administrator's handbook. Jeffrey L. Harris, 2002.

[112] Susan Powers. IBM Server iSeries handbook, version 5 release 1. 22nd ed. IBM, International Technical Support Organization, 2001.

[113] David Wai-lok Cheung. Open Source Software and its impact to Technology Development. E-Commerce Strategies for Development, 2003.

[114] The Portable Application Standards Committee, http://www.pasc.org/. 2004.

[115] TZ JI. OS/2: features, functions, and applications: standard edition 1.0. New York: Wiley, 1988.

[116] Netware Standards Sub-Committee. Netware Administration Standards. John Hopkins Institutional Computing Standards, 2003.

[117] IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York, NY: 1990.

[118] Citeseer Scientific Literature Digital Library, http://citeseer.nj.nec.com/. 2004.

[119] Gartner Inc. WFS Financial Support Study. 2002.

[120] Cybersource® Pty. Ltd. Linux vs. Windows: Total Cost of Ownership Comparison. 2002.

[121] Al Gillen, Dan Kusnetzky, Scott McLarnon, Scott McLarnon, Randy Perry. Linux and Intel-Based Servers: A Powerful Combination to Reduce the Cost of Enterprise Computing. 2002.

[122] Meta Group. The Impact of OS/Platform: Selection on the Cost of ERP, Implementation, Use and Management. 2002.

[123] Lorraine L.Cosgrove, Christian A.Christiansen. Server Selection: Reversing the Trend of Rising IT Costs. International Data Corporation, 2002.

[124] CIOview Corp. A Business Case for Windows Server Optimization. 2002.

[125] Netcraft Services; http://www.netcraft.com. 2001.

[126] http://www.netsys.com/. 2004.

[127] Computing and Information Technology Interactive Digital Educational Library (CITIDEL), http://www.citidel.org/. 2004.

[128] http://www.allconferences.com/. 2004.

[129] Apple Corporation: http://www.apple.com. 2004.

[130] Hewlett Packard Corporation, http://www.hp.com. 2004.

[131] Rebecca Buckman. Face-Off over People's PCs. The Wall Street Journal Online 2003.

[132] James Gray. The State of Linux. Linux Journal 2003.

[133] Paul L.Stephenson, Neal T.Rogness, Justine M.Ritchie, Patricia A.B.Stephenson. SPSS Manual for Moore and McCabe's Introduction to the Practice of Statistics. 3rd ed. W H Freeman & Co, 1998.

[134] David G.Kleinbaum, Lawrence L.Kupper, Keith E.Muller, Azhar Nizam. Applied Regression Analysis and Multivariable Methods. 3rd edition ed. Duxbury Press, 1997.

[135] Richard A.Johnson. Applied multivariate statistical analysis. Prentice Hall, 2002.

[136] StatSoft I. Multiple Regression, http://www.statsoftinc.com/textbook/stmulreg.html. StatSoft, Inc., 2004.

[137] G.A.Moore. Crossing the Chasm. New York: Harper Business, 1999.