Modeling the Evolution of Operating Systems: An Empirical Study

YI PENG[1], FU LI[2] AND ALI MILI[1]
New Jersey Institute of Technology, University Heights, Newark, NJ, 07102, USA
(email: {yxp1898, ali.mili}@njit.edu, fxl9575@ADM.njit.edu)

SUMMARY

In this paper, we report on an empirical experiment where we observe, record and analyze the evolution of selected operating systems over the past decades and derive a statistical model that captures relevant evolutionary laws. We use this model to highlight relevant statistical laws, as well as to predict the future evolution of operating systems. In addition to deriving predictions on the future of specific operating systems, we also focus on predicting emerging features of operating systems, thereby obtaining a profile of future operating systems.

KEY WORDS: Operating Systems, Software Engineering Trends, Tech Watch, Unix, Solaris, Sun/OS, BSD, Windows, MS-DOS, MAC OS, Linux, Net Ware, HP-UX, GNU Hurd, IBM AIX, Compaq/ DEC VMS, OS/2.

1. The Evolution of Operating Systems

Operating systems have played a crucial role in the evolution of the computing field, adapting continuously to the evolution of hardware and to the evolving demands of user communities. They have been the subject of intense research since the early days of computing, and have given rise to some of the most important breakthroughs in the history of computing. In this paper, we report on an empirical experiment where we observe, record and analyze the evolution of selected operating systems over the past decades and derive a statistical model that captures relevant evolutionary laws.

This work is part of a wider project whose purpose is to model and understand the evolution of software engineering trends [1]. In addition to the top down, deductive, approach advocated in [1], we have resolved to take a bottom up, inductive, approach, whereby we analyze / model evolutionary laws of specific families of trends, to subsequently derive more generic laws. In [2], Chen et al discuss the evolution of programming languages; in this paper, we apply the same methodology to another homogeneous family of products, namely operating systems. Operating Systems are an excellent test bed for our inductive approach, for the following reasons:

[1] Department of Computer Science

[2] Department of Mechanical Engineering

- Operating systems offer homogeneity of purpose, function and structure, across several decades of evolution, thereby affording us a meaningful analysis.
- Operating systems have played and continue to play a central role in the evolution of computing; hence understanding their evolution elucidates that of the field of computing in general.
- Operating systems offer a wide diversity of features and a long historical context, thereby affording us precise analysis of the relations between their characteristics, their context, and their evolution.
- The history of operating systems is relatively well documented, and their important characteristics relatively well understood, thereby facilitating data collection and data analysis.

In section 2, we introduce the set of operating systems that we have selected as our sample, and briefly discuss the factors that we have used to characterize them, for the purpose of our statistical analysis. In sections 3 and 4 we discuss in turn, how to quantify (i.e. assign a numeric function) the factors we selected, then how to collect data on these factors (i.e. compute the numeric functions associated with each factor). In section 5 we briefly introduce our raw data, then discuss the main results of our statistical analysis of this data. In section 6, we derive a predictive model from our statistical results, and use it to predict the future evolution of existing operating systems, as well as the future evolution of operating system features, culminating in a tentative definition of the profile of future operating systems. In section 7 we conclude by summarizing our findings, discussing their statistical validity, and sketching venues for further research.

2. Characterizing Operating Systems

2. 1 Sample Operating Systems

For the purposes of our statistical study, we have selected a set of fifteen operating systems: Unix, Solaris/ Sun OS, BSD, Windows, MS-DOS, MAC OS, Linux, Net Ware, HP UX, GNU Hurd, IBM Aix, Compaq/ DEC VMS, OS/2. We have selected these operating systems to meet the following broad coverage criteria:

- To cover a wide range of periods in the history of operating systems/ computing.
- To cover a wide range of platforms on which these operating systems run, representing successive technological innovations.
- To cover a wide range of pedigrees, from large operating systems with heavy corporate backing to small operating systems started by individuals or small teams.
- To cover a wide range of design philosophies, from layered design, to hierarchical design to client server design.
- To cover a wide range of design features, such as diverse scheduling policies, memory management policies, user interfaces, etc.
- To cover a wide range of functionality, from simple user support to sophisticate fully integrated environments.

- To cover a wide range of openness levels, from open source software to highly integrated commercial software.
- To cover a wide range of innovations in the design of operating systems (virtual memory, time-sharing, paging, networking, etc).

Given our sample of operating systems, we must now characterize each operating system by means of relevant, general-purpose, factors. We distinguish between two types of factors, which we discuss below: intrinsic factors, and extrinsic factors. These factors have been selected on the basis of three broad criteria: relevance (our belief that they affect the fate/ evolution of an operating system); general significance (our belief that they are meaningful for all the operating systems in our sample); orthogonality (our belief that they reflect distinct aspects of operating systems). In [3] we give, for each factor, a brief definition and a brief discussion of why we feel this factor affects the evolution of operating systems.

2.2 Intrinsic Factors

The intrinsic factors of an operating system are factors that characterize the operating system *per se,* independent of any environmental context; in particular, intrinsic factors are time-independent. For the purposes of our study, we have identified nineteen (19) intrinsic cost factors, which we have divided into seven broad categories; these are briefly introduced below.

2.2.1 Resource Management

We have identified four intrinsic factors that fall under the category of resource management.

1. Scalability: Scalability [4] is an operating system's ability to increase its processing capacity as CPUs are added. This is very significant factor of operating systems as modern multi-processor computing get more and more popular.

2. CPU Management: CPU management [5] is an operating system’s ability to manage CPU cycles. With the evolution of hardware, the focus of CPU management has evolved from maximizing throughput to minimizing response time, but has remained important throughout.

3. Memory Management: Memory Management [5] pertains to the operating system’s ability to manage its memory hierarchy. Like CPU management, this factor has remained important despite the evolution of hardware, with its focus shifting from maximizing throughput to minimizing response time

4. I/O management: The factor of I/O management [5] reflects the design choices made by the operating system, and their impact on the system’s functional and operational properties. I/O is the bottleneck of any operating system, and its management plays a crucial role in the system’s performance.

2.2.2. Usability

The usability of operating systems reflects the ease with which users can learn and use the system. We have identified three dimensions of usability, which we capture in the following intrinsic factors; we admit that these factors are not perfectly orthogonal, but we still separate them as they are distinct aspects of usability.

1. Ease of Learning: Ease of learning [4] can be quantified by the number of hours required to acquire a predefined level of proficiency. This is clearly an important factor since it may directly affect the degree of popularity of the operating system.

2. Ease of Use: Whereas [4] ease of learning reflects the effort required to learn an operating system, this factor reflects the ease of using it once one has learned it. Whereas ease of learning may affect an operating system's ability to attract users, this factor affects its ability to keep them.

3. Consistency of Interaction Protocols: This factor reflects the uniformity of user commands and operating systems outputs, and has the potential to greatly enhance ease of learning and ease of use [4].

Whereas usability reflect the ease of use of an operating system, usefulness, which we discuss in the sequel, reflects the level of service provided by the operating system. We distinguish between two broad aspects of usefulness: functional and operational.

2.2.3. Functional Usefulness

We have identified five factors of functional usefulness, which we introduce in turn below.

1. Range of Services: This factor reflects the range of functions offered by the operating system [6]. This factor clearly plays an important role in characterizing an operating system, hence possibly in affecting its fate.

2. Extent of Programming Languages Support: One aspect of functional usefulness of an operating system is its ability to support programming languages; rather than merely counting how many languages an operating system supports, which would be based on the faulty assumption that all languages are equally important, we have chosen a predefined set of languages. These are: Ada, ALGOL, Pascal, C, C++, COBOL, FORTRAN, Java, Perl, LISP. Each operating system is judged by how many of these it supports.

3. Distributed Computing: This factor reflects the ability of an operating system to support applications that run on geographically distributed sites or process data in geographically distributed sites [6]. This factor is clearly important in today's computing landscape, where supporting distributed applications is a prerequisite.

4. Network Services: Network services are provided to support distributed applications requiring data access and applications interoperability in heterogeneous or homogeneous networked environments. In the era of pervasive networking, network services are very important for an operating system [6].

5. Deadlock Management: This factor reflects the policy adopted by the operating system to deal with deadlocks [5]. While we do not expect this factor to have a great impact on the usefulness of the system, we are including it for the sake of completeness.

2.2.4. Operational Usefulness

We have identified two operational aspects of usefulness of an operating system: reliability and security.

1. Reliability: Reliability is the ability of a system to perform its required functions under stated conditions for a specified period of time [4]. Reliability is generally considered critical by end users, who depend on operating systems for many important tasks.

2. Security and protection: Security and Protection is the ability of a system to manage, protect, and distribute sensitive information [4]. End users generally place a great premium on security, since operating systems can be custodians of much sensitive information.

2.2.5. Versatility

Versatility of an operating system is its ability to run on a wide range of platforms, under a wide range of distinct circumstances. We have identified three aspects of versatility, which we discuss below.

1. Portability: This factor reflects the ease with which an operating system can be transferred from one hardware or software environment to another [4]. This factor is clearly important, as it has an impact on how widely used an operating system can be.

2. Compatibility: This factor reflects the ability of an operating system to adapt to environmental conditions that were not specifically design for it [4]. The archetypical example of compatibility is compatibility of a version with respect to earlier versions of the same operating systems; we could imagine broader examples as well. This factor is important because it affects the system's ability to gain a foothold in new markets, for example.

3. Openness: Openness is the degree to which an operating system complies with open standards [4]. Clearly, openness enhances an operating system's chances of widespread acceptance.

2.1.6. Design

This factor reflects design qualities of the operating system, such as integrity, economy of concept, orthogonality, adherence to design principles, etc. Even though the design of an operating system is not directly palpable to the user, in the way other intrinsic factors are, we feel it may play an important role in characterizing the evolution of an operating system because it may affect nearly all the other intrinsic factors [7].

2.1.7. Cost

This factor considers acquisition costs, maintenance costs, and operating costs of an operating system [4,6]. Though perhaps to a limited extent, we anticipate that the attending costs of an operating system may influence its evolution.

2.3 Extrinsic Factors

Whereas intrinsic factors characterize the operating system per se, extrinsic factors characterize the environment in which the operating system evolves. By their very definition, extrinsic factors are time-dependent. In our statistical model, extrinsic factors act as independent variables and as dependent variables: past values of these factors act as independent variables, and future values act as dependent variables. As we will se when we introduce them, each of these factors is likely to affect its own evolution (its past values affect its present/ future values) as well as the evolution of others (the past values of one affect the present/ future values of others). We represent the status of an operating system by the vector of its extrinsic factors. The extrinsic factors that we have selected for our study are introduced below.

2.3.1 Institutional support

The factor of institutional support reflects how much support the operating system is finding in academic institutions and research laboratories. Specific questions include:

1. Support the OS: the institutional unit provides the environment for a given operating system and allows people to use 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 the operating system in one or more courses.
4. Research using the OS: in the institutional unit, the operating system is used in research activities.
5. Research on the OS: in the institutional unit, the operating system is the subject of research.

2.3.2 Industrial support

The factor of industrial support reflects the amount of support the operating system is getting in industry. Levels of support are coded as follows:

1. No Support for the operating system: the industrial unit does not provide the environment for the operating system and does not expect employees to use it.
2. Support using the operating system: the industrial unit provides the environment for a given operating system and allows employees to use it.
3. Encourage using the operating system: the industrial unit encourages the usage of the operating system within the unit and provides the necessary infrastructure (environments, platforms, technical expertise, etc).
4. Require using the operating system: the industrial unit requires the usage of the operating system to support the unit's activities.

2.3.3 Governmental support

The factor of governmental support reflects whether and to what extent the operating system is supported by a governmental agency. Levels of support are coded as follows:

1. No support for the operating system: the governmental unit does not provide the environment for the operating system and does not expect employees to use it.
2. Support the operating system: the governmental unit provides the environment for a given operating system and allows employees to use it.
3. Encourage using the operating system: the governmental unit encourages, but does not mandate, the use of the operating system within the unit.
4. Require using the operating system: the governmental unit requires the use of the operating system to support the unit's activities.

2.3.4 Organizational support

The factor of organizational support reflects the support of professional organizations for this operating system. Specific questions include:

1. Is this operating system introduced and/or supported by any (international) organization?
2. Are there any organizational standards?
3. How many conference series are devoted to this operating system?
4. How many conference papers/articles are published on this operating system?
5. How many conference papers/articles are published using operating system?

2.3.4 Grassroots support

The factor of grassroots support reflects the support of professionals and practitioners for this operating system. For a given operating system, we want to determine the proportion of professionals/ practitioners who: know this operating system; use this operating system, among others; work exclusively in this operating system.

3. Quantifying Characteristic Factors

In order to conduct a statistical analysis of our sample of operating systems, we must quantify all the factors we introduced in section 2, i.e. assign them a numeric function. We distinguish between the quantification step (which is represented by the question:

what numeric function reflects this factor?) and the evaluation step (which is represented by the question: how do we compute the selected numeric function). In this section, we focus exclusively on the quantification step, and discuss evaluation in section 4. We have divided the set of factors into five different categories, which correspond to five distinct quantification methods; we discuss these below.

3.1 Numeric Factors

This family contains all the factors (intrinsic or extrinsic) which are numeric by definition (e.g. acquisition cost for an operating system, measured in dollars, for US delivery), or for which a numeric formula is well known and widely accepted (e.g. MTTF for reliability). Other factors that fall in this category include the extrinsic factors, for which we measure the proposition of practitioners that correspond to each level of support (for example, 20 percent of industrial organizations do not support operating system X, 80 percent support it, 60 percent encourage it, and 25 percent mandate it).

3.2 Hierarchical Factors

We place in this category factors that can be characterized by a predefined set of features, where the features are ordered (e.g. represent increasingly sophisticated implementations of an operating systems function). This order is not necessarily total, since some features may be deemed equally sophisticated. We assign ranks 1 (least sophisticated) to N (most sophisticated) to the features, where N is the number of ranks defined among the features. The score of an operating system is then derived as the sum of all the scores that corresponds to the features it has.

For example, in order to quantify memory management, we take 11 sub-factors into consideration, and range them from garbage collection (score:1) to shared-memory multiprocessor (score:6), as show in Table 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, we will look through all these sub-factors, decide all the provided factors, and derive the final score for this operating system by summing up the scores of all the supported sub-factors.

Table 1 Hierarchical Sub-feature Quantification: 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 openness, we have looked through a list of 23 standards that could be implemented by operating systems. Here is the list:

Letter Representation	Open Standard
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, we have constructed the hierarchy of these standards, as shown in Figure 4.2. From this figure, we know 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.

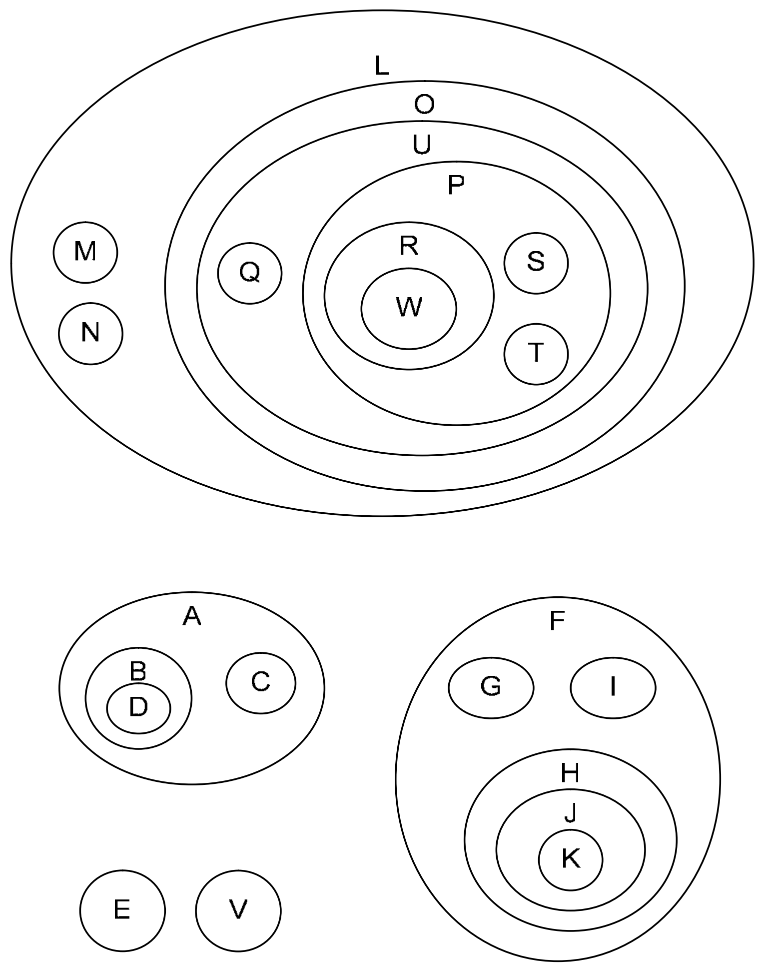

Figure 1 Standards Hierarchy for Openness.

By analyzing the standards hierarchy, we have assigned different scores for different standards according to their position in the hierarchy, as shown in Figure. 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 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.

We acknowledge that this method may sound controversial as it appears arbitrary. But we argue that it is adequate for our purposes, as it generally reflects our intuition about how candidate operating systems stand with respect to the intrinsic factors; also, we argue that any effort to quantify what are essentially qualitative attributes may appear to be arbitrary somewhere.

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

3.3 Cumulative Factors

This class of factors is a special case of the previous class, in which all the features of interest have the same rank, say 1; whence application of the formula for hierarchical factors produces the number of features that the operating system has. Two factors fall in this category: *Programming Language Support*, and *Consistency of Interaction Protocols.*

3.4 Weighted Factors

This category of factors can be viewed as a special case of cumulative factors, where the score of an operating system for a given feature is not merely 0 or 1, but ranges over a discrete five-value scale (ranging from 1 for poor to 5 for excellent), depending on how the operating system fares with respect to each feature. For example, eight features are used to quantify the factor of ease of use. To each feature, we assign a score between 1 and 5 depending on how well the operating system does with respect to this feature. The overall score of the operating system is the sum of all eight scores. This category of factors includes ***ease of use*** and ***design***.

Table 2 Quantifying Methods

Factor	**Quantifying Methods**			
	Numeric Factors	Hierarchical Factors	Cumulative Factors	Weighted Factors
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	√			

4. Computing Characteristic Factors

In section 2 we classified our relevant factors in terms of their definitions / significance, and in section 3 we classified these factors in terms of their quantifications. In this section we classify them in terms of how we evaluate them. Conceptually, at least, these three classifications are orthogonal, i.e. they cut across the set of factors independently of each other.

4.1 Intrinsic Factors

Because most of the intrinsic factors are technical-centric, stable and usually well documented, we gather the data from non-survey resources such as operating system textbooks, technical papers, systems manuals, journals and other similar documents. The resources we choose are authoritative and well-known [5,6,8]. The following intrinsic factors are collected in this manner: CPU Management, Memory Management, I/O management, Openness, Scalability, Consistency of Interaction Protocol, System Services, Range of programming languages, Distributed Computing, Network Services, Deadlock management, Security & Protection Management, Portability, Compatibility, Cost, Organization Support. Below we show operating systems' openness scores as an example.

Table 3: Openness Score

OS	Openness Score
UNIX	47
Solaris/Sun OS	47
BSDs	35
OS/2	27
Windows	30
MS-DOS	1
MAC OS	41
Linux	40
NetWare	22
HP-UX	46
GNU Hurd	36
IBM AIX	47
Compaq/DEC VMS	46

4.2. Grassroots Factors

A number of factors are not purely technical features, for example, institution support, governmental support, etc. These factors can not be decided by analyzing its technical merits. Therefore, we do the data collection via survey pages on our survey website.

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, we post survey invitations 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, comp.os.ms-windows.nt.misc, etc. We choose these operating related websites, 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, we have about 800 records gathered from the survey websites. Figure 1 is a screen shot of the web survey page. Also, we demonstrated our research to high school and college students and invited them to participate in our web survey. In order to attract more users, we also posted several announcement messages to NJIT web sites, inviting students to participate in the survey.

5. Statistical Analysis

Factor analysis is used to draw some initial conclusions on raw data. In this project, factor analysis is used to investigate the latent factors in intrinsic and extrinsic factor groups. Canonical analysis is used as an advanced stage of factor analysis.

5.1 Statistical Variables

Orthogonally to all the classifications we have made so far of our quantitative factors, we make one more classification, pertaining to their role in our statistical analysis.

- Independent variables. We let the independent variables of our model be the set of intrinsic factors as well as the past histories of each extrinsic factor.
- Dependent variables. We let the dependent variables of our model be the current or future values of extrinsic factors.

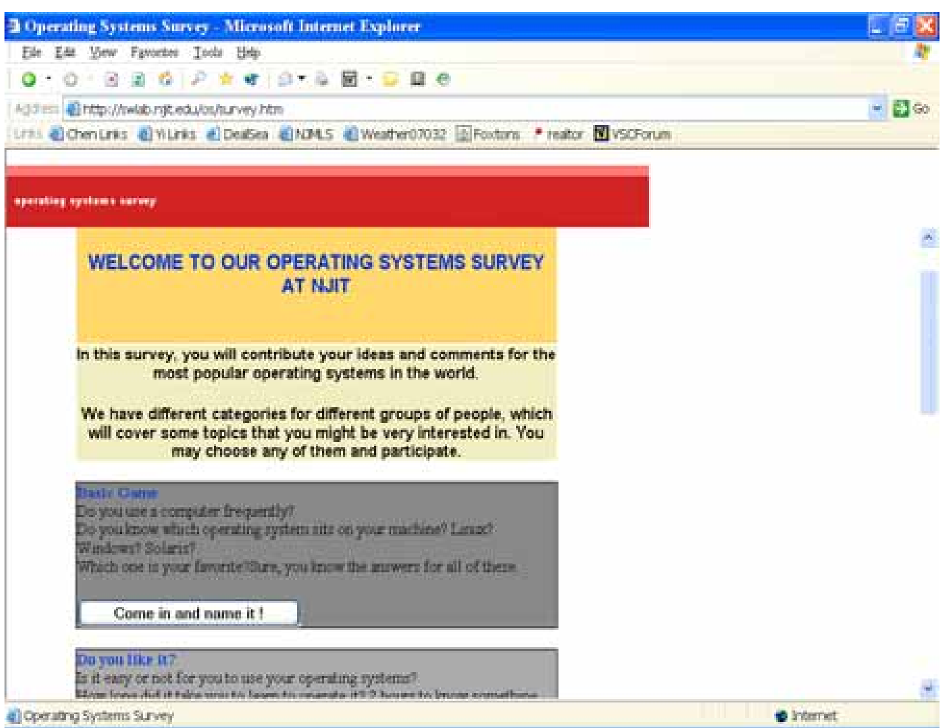

Figure 1: Operating system survey website

In light of this classification, the purpose of our analysis is to highlight statistical relations between the intrinsic factors and past extrinsic factors on one hand, and the present/ future extrinsic factors on the other hand. If we posit that the status of an operating system is represented by its vector of extrinsic factors, then the statistical model will allow us to determine how its intrinsic properties and its past evolution determines its future evolution.

5.2 Raw Data

In this section, we present the raw data collected for independent variables, both intrinsic factors and extrinsic factors. In Table 4, we present some final score for some intrinsic factors.

Table 4: Intrinsic factor samples for operating systems

OS	Scalability	CPU Management	Consistency of Interaction Protocols	System Services	Programming Languages Supported
UNIX	28	36	10	47	10
Solaris/Sun OS	28	36	12	51	10
BSDs	21	28	10	51	10
OS/2	15	6	8	51	9
Windows	28	28	12	51	10
MS-DOS	6	1	9	41	8
MAC OS	21	28	12	51	10
Linux	28	36	12	51	10
NetWare	15	21	10	51	10
HP-UX	21	28	12	51	10
GNU Hurd	21	28	8	40	9
IBM AIX	28	36	12	51	10
Compaq/DEC VMS	21	21	12	51	10

In Table 4, we present the institutional support data as a sample for the raw data we collected in our project. The following Figure shows the operating systems' institutional support score in 2003. We can see that Windows, Solaris and UNIX have higher institutional support than others. There are many detailed results available on our web survey at: http://swlab.njit.edu/os/survey.htm. For every category for the survey, we collect about 800 records from response. The exact number differs from category to category, because each responder replies to only selected/relevant questions.

The full data set is available in [3], as it showed in June 2004. The data is updated usually as operating system professionals fill out the survey forms

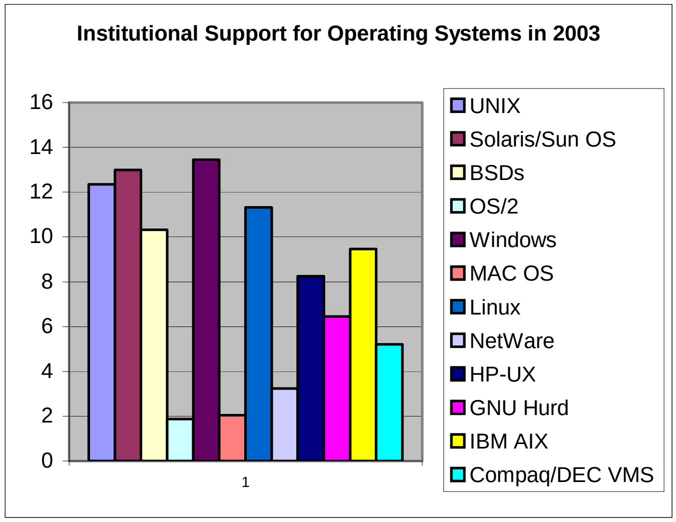

Figure 2: Institutional support for operating systems in 2003

5.2 Factor Analysis

Principal Component Analysis (PCA) [9] 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. We try to reach the following goals:

- Reduce the number of components to be considered;
- The extracted components should preserve most of the relations with the independent variables.

Table 5: Factor Analysis for Intrinsic Factors

Component	Eigenvalue	% 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

In Table 5, the result of factor analysis is shown. From this table, we can see that 6 components can cover 95.903% of the variation in dependent variables. Hence for all intents and purposes, the eleven intrinsic factors we have selected represent a space of dimension six rather than 19. Thus, we extracted 6 components from our initial 19 intrinsic factors. From another perspective, as Table 6 (Rotated Component Matrix) shows, we could also see that 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 we had listed at the beginning of this section.

Table 6: Rotated Component Matrix

OS	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

5.3 Canonical Correlations

We also use canonical correlation as an additional procedure for assessing the relation-

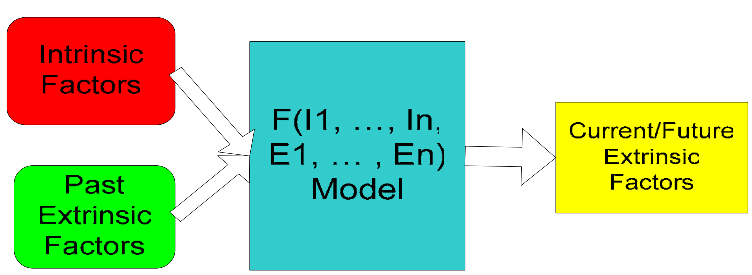

Figure 3: Regression Model for Operating System Trend

ships between independent variables and dependent variables. In this step, Pearson correlation analysis [10] is used to analyze this relationship. 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. Table 7 shows the results of correlation analysis of government support and intrinsic factors.

Table 7: Correlation Analysis

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

From Table 6 we can see that government support is affected by all the intrinsic factors to some extent. Yet, from the perspective of which one is the most significant, security protection, scalability and design have considerable weight in winning government support. The correlation analysis step 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.

6. A Predictive Model

6. 1 Regression Model

After factor analysis has been done, we discuss how to construct regression models by using the intrinsic and extrinsic variables. The multivariate regression [11] is used to construct the models. The multivariate regression equation has the form:

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

Where:

Y = the predicted value on the DV,
α = the Y intercept, the value of Y when all X s 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)

The multivariate regression model for operating system trends consists of 5 parts, one for each dependent variable. Due to the number of factors used in the model, there will be a balance between model reliability and the information completeness. The more factors we use, the more information is included, but the regression model is expected to be less reliable. The fewer factors we use, the less information is included, but the regression model is expected to be more reliable. We use SPSS to calculate the parameters for this model.

The factor analysis shows that the six extracted components are sufficient to account for more than 95.903% of the variance of the independent variables. To construct useful regression model for the historical trends, the components, instead of original factors, will be used. From the factor analysis, we find that intrinsic variables do represent important features of an operating system.

Multivariate regression model is a useful approach to predict the future trend based on existing data. It is our goal to adopt the regression model and apply it to the specific needs of analyzing the trend of an operating system.

SPSS is used to construct the multivariate regression model, which uses the historical independent extrinsic factors and six independent components that extracted from intrinsic factors as input independent variables. The SPSS output will show the regression model for each extrinsic factor. The parameter will show the impact of each input independent factor to the output dependent variable. By using all of the parameters, the regression models are constructed for operating system trends.

6.2 A Predictive Model for Operating Systems

In order to predict the future trends of operating systems, the original multivariate regression models should be revised. The derivative regression model will show the relationships among data of 1997, 2000, and 2003. Derivative regression models are constructed as follows:

$$E_{2003} = A*I + B*E_{2000} + C*E_{1997} + D$$

Where:
E2003: Value of extrinsic factors in 2003
I: Value of intrinsic factors
A: Coefficient for intrinsic factors
E2000: Value of extrinsic factors in 2000
B: Coefficient for extrinsic factors in 2000
E1997: Value of extrinsic factors in 1997
C: Coefficient for extrinsic factors in 1997
D: Constant value to adjust for the unknown factor

We use least square as the criterion to judge whether the regression model converges. SPSS is used to calculate the parameters 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.

6.3 Predictive Model: Operating System Features

In addition to predicting the future of individual operating systems, we feel that it is important to try to predict the evolution of individual operating systems features as well, for three reasons:

- The fate of an individual operating system is subject to many external factors that are not reflected in our quantitative model: for example, an operating system may be superseded by a more powerful/ more efficient/ less costly version; or an operating system may lose the support of a major user organization, a major provider, a major standards organization, etc. By contrast, the evolution of a particular feature is continuous, in the sense that it is much more immune to sudden individual decisions.
- New operating systems may emerge in the future, altering the landscape and rendering our predictions irrelevant. The emergence of new operating systems does not alter our prediction of operating system features, however. We expect that any emergent operating system will have the features that our model finds to be important in the future.
- The prediction of operating system features bears significance on its own, as it helps us sketch the profile of future operating systems, by showing which features are likely to emerge as important, and to what extent.

The success of an intrinsic factor can be measured by the extent to which this factor is correlated with operating system success: whenever this factor is preponderant in an operating system, the system is successful, and vice versa. The preponderance of an intrinsic factor, say IF, in an operating system, say os, is measured naturally the score that the operating system has for that factor, say IF(os). As for the level of success of an operating system, it depends on two parameters: First, which extrinsic factor we use to quantify success; second, at what year are we measuring this factor; we denote by EF(os,Y) the score obtained by operating system os for extrinsic factor EF at year Y. The degree of success of a given intrinsic factor IF at year Y with respect to the extrinsic factor EF can then be quantified by the statistical correlation of IF(os) with EF(os,Y) for all operating systems os. We write this formally as

$$P_{IF,EF}(Y) = \underset{os \in \Omega}{Cor}(\text{IF(os), EF(os,Y)})$$

Where:

Cor() is Pearson's statistical correlation;

$P_{IF,EF}(Y)$ stands for preponderance of IF with respect to EF at year Y.

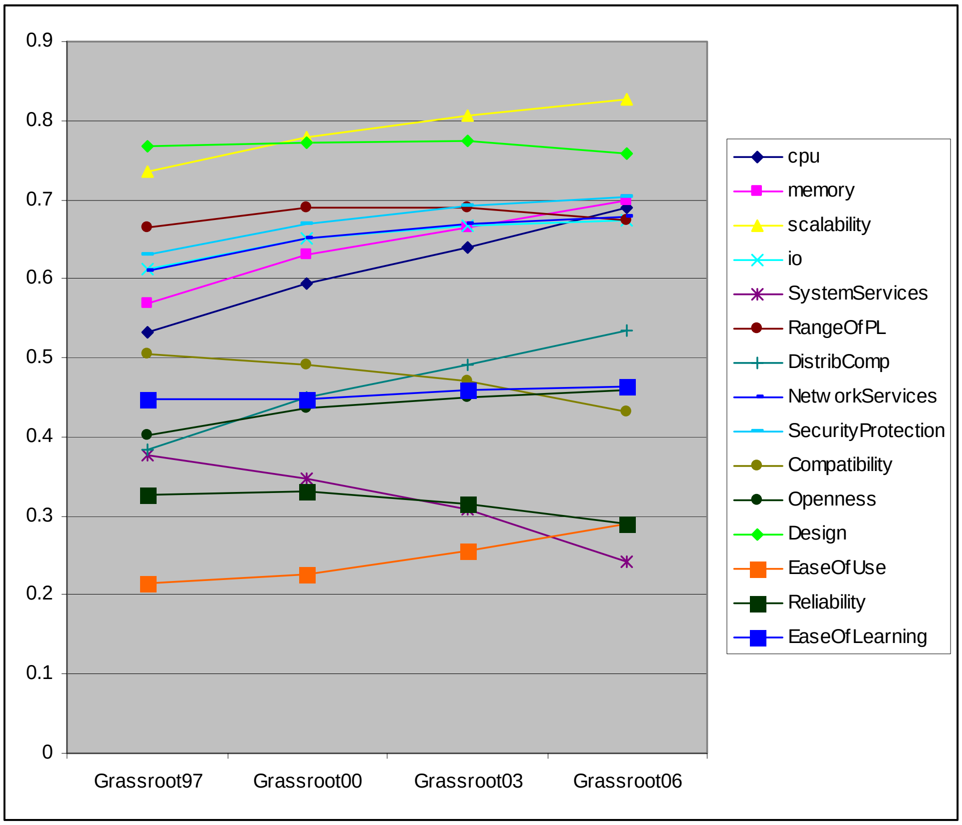

Figure 4 Preponderance of some IF's for EF = Grassroots Support

Using this formula, we can compute the preponderance of any intrinsic factor throughout the scope of our experiment (1997, 2000, 2003). Perhaps more interestingly, we can use the predictive model derived in section 6.3 to extrapolate the preponderance of any intrinsic factor in the future.

According to the concept of preponderance for a given intrinsic factor respected to an extrinsic factor in that particular year, we construct Figure 4 as an example. The listed intrinsic factors and an extrinsic factor --- Grassroots supports are selected. One line is drawn for every intrinsic factor during different years.

6. Conclusion

6.1 Summary

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 this paper, we propose to model the evolution of operating systems by quantifying their intrinsic attributes and their environmental conditions, and highlighting statistical relationships between them. Then we have collected quantitative data about intrinsic attributes using operating systems' literature and have colleted quantitative data about environmental attributes using an online survey.

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, we also want to analyze the evolution of operating system features. So that, even if we can not tell what operating system would be successful or unsuccessful, we could at least characterizing future of operating systems by their main attributes.

6.2 Statistical Validation

After the statistics models for operating system evolution have been constructed, a number of methods need to be introduced to assess the validity of these models. To this effect, we have computed the F-Statistic value of many of our tables of results, these can be found [3]. As an example, we show below to the table that quantifies governmental support for 2003, according to the predictive model, and according to our collected data. A mere inspection shows that these tables are very similar: the F-Statistic value for this table is found to be F = 0.014, which is deemed low enough to prove validity.

Table 7: 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

F-Statistic [10], which is a standard statistical method to check if there are significant differences between groups, is used to validate the prediction.

7.3 Prospects

The empirical study for modeling software 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.

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 follow the similar pattern of analysis. Generally, empirical method will be used first. After having better understanding of trends behavior, analytical method will be applied to understand the cause/effect relationships.

References

[1] Robert David Cowan, Ali Mili, Hanny Ammar, Alan McKendall Jr., Lin Yang, Dapeng Chen et al. Software Engineering Technology Watch. IEEE Software 2002;123-130.

[2] Y.Chen, A.Mili, L.Wu, R.Dios, K.Wang. Programming Language Trends: an Empirical Study. submitted to IEEE Conference 2003.

[3] Yi Peng. Characterizing the Evolution of Operating Systems; PhD Dissertation. 2004.

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

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

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

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

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

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

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

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