

Linux vs. Windows

Software Engineering Perspective

Mohammad Anas Ramadan

The Arab Academy for Banking and Financial Sciences
anas at linuxfuture.org

ABSTRACT

So far, there is a lot of publications and articles try to compare between Windows and Linux from some specific aspects and features, which doesn't look like a well-structured aspects. This will lead to several questions, how these aspects have been chosen? And can I say that these aspects are necessary and enough to lead me to the right judgment decision with respect to the balance between Windows and Linux?

In order to answer this question, and as an attempt to reach the right judgment, I'll try to define these 'aspects' first, depending on software engineering perspective, then I'll start compare between Windows and Linux depending on them.

Keywords

Linux, Windows, desktop, Linux vs. Windows, Software Engineering, Debian, Slax.

INTRODUCTION

There are three questions, should be answered first:

- What distribution of Linux should be chosen to compare?
- What version of Windows should be chosen?
- What are the exact comparison factors should be used?

Since the objective here is to compare between the capabilities and the features of Windows in general, and Linux in general, not as (specific products), finding answers will not be easy.

After that, we can start compare between windows and Linux according to a well identified software comparison factors.

LINUX DISTRIBUTIONS

According to DistroWatch.com, there are more than 350 distributions (flavors) of Linux; most of them are derivative distributions¹. While a little number of them are 'root distributions', this means: developed from scratch, which means that they were not derived from an already exist distribution.

The question here is how can one distinguish between root distributions and derivative distributions?

Operating System, at the end of the day, is a set of files, can be expanded or reduced efficiently by a software called "Package Manager" that understands the operating system's file structure, and conduct the operation of adding/removing file. So that we can consider it as the "Backbone" of the operating system file structure, and thus, the backbone of the operating system itself.

And, thus we can consider Package Manager as the essential factor in distinguishing between the root distributions themselves, and between the root and the derivative distributions.

If a distribution has been developed by organizing components/applications in a new backbone/package manager developed from scratch, this distribution will be considered as a root distribution. If it has been developed by organizing components/applications using an already exist backbone/package manager, which means an already exist file structure, and thus an already exist operating system, it will be considered as a derivative distribution.

As another classification factor, beside the package manager, Desktop Environment will be another one, since in Linux; there are two completely-different desktop environments for Linux: GNOME, and KDE.

According to these two factors, most common Linux distributions can be organized in a matrix like this:

		Desktop Environment		
		GNOME	KDE	
Root Distributions (The Package Manager)	Debian	Deb	Debian - Ubuntu	Knoppix - MEPIS
	Slackware	TGZ	Nonux - Topologilinux	Slackware - Slax
	Redhat	RPM	RedHat	SUSE - Mandriva
	Gentoo	SRC	Gentoo – Ututu	Kororaa

The difference between the root distributions: How to add applications – the Package Manager – The “Extension/Scalability Gate”

The difference factor between distributions derived from a specific root distribution: What to add?

In order to cover Linux in general, I had to cover two distributions at least: a GNOME-based distribution, and a KDE-based distribution.

Regarding the package manager, Gentoo is not recommended for end users since its package manager is directed only for expert users. RPM also has a bad reputation—every time user adds an RPM software package, system will ask him/her for another one.

Ubuntu is a Debian-based distribution, and it is the most popular distribution all over the world nowadays, it has a very powerful package manager, and offers GNOME desktop natively.

Slax is a Slackware –based distribution, its package manager is pure simple, without dependency check. It's KDE-based distribution, and it's distributed as a live CD.

So I'll focus on both Ubuntu and Slax.

WINDOWS VERSIONS

Windows 9x	Windows NT
Windows 95	Windows NT 4.0
Windows 98	Windows NT 5.0
Windows 98 SE	Windows 2000
Windows ME	Windows XP
	Windows Vista

Simply I'll choose Windows XP, since it mostly covers all features and functionality afforded in previous versions of Windows. Despite the fact that Windows 9x series is still considered as a better choice from performance aspect, while Windows Vista is not affordable yet.

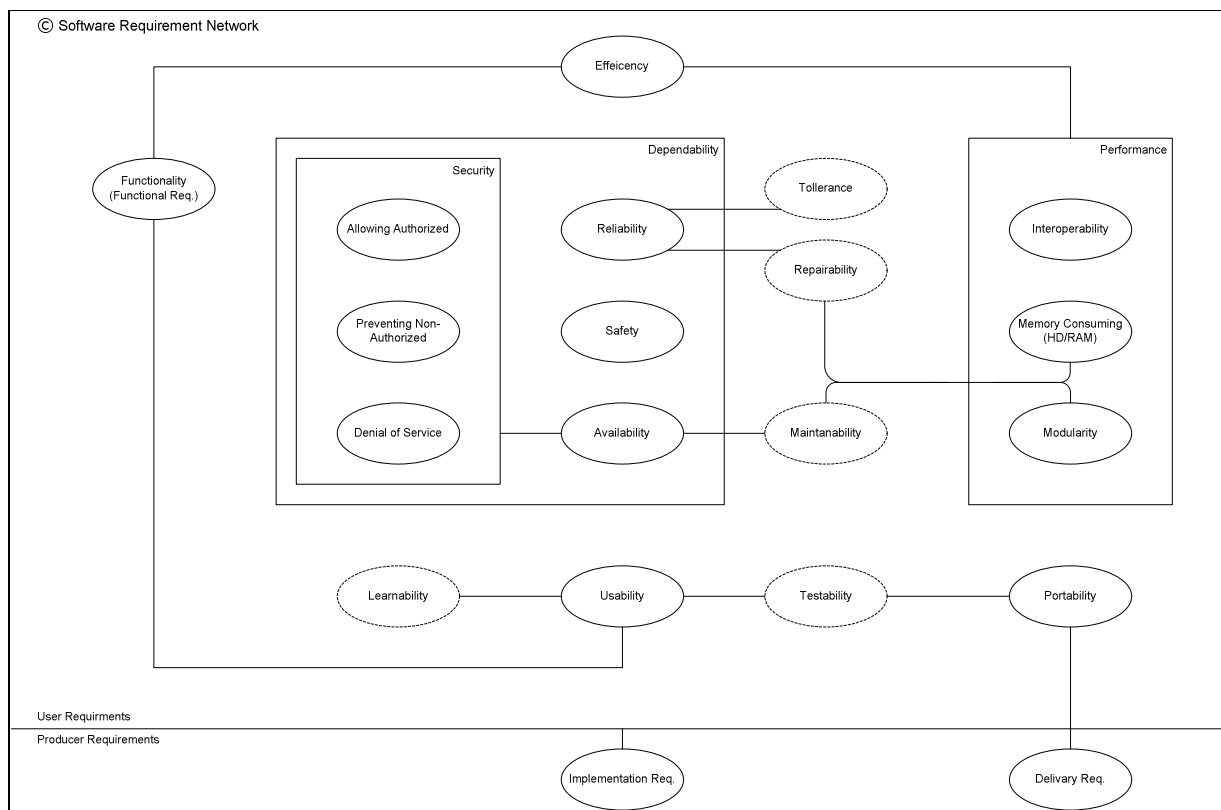
COMPERISON FACTORS

As it's mentioned above, studies that try to compare between Windows and Linux, does not start from a structured comparison factors, and actually that's due to the fact that:

- There's no common measure to evaluate software, this due to the intangible nature of software, aside from the fact that software development is still considered as a creative process.
- Software Engineering seems like it developed by "the commercial software community" and there's no strong enough contribution in this field from open source community.

What I'll try to do here is to identify and structure these factors first, which will help in bringing researcher to a better view of each factor, and realizing how can each affects the rest– the reason that makes evaluating software very complex procedure, then I'll start compare according to these chosen factors.

Simply, I will compare how far both Windows and Linux meet software requirements.



FUNCTIONAL REQUIREMENTS

Functionality, or functional requirements usually depends on the system, each system is designed for offering performing specific functions. But since we are talking here about operating systems, we can consider that they should cover the basic four aspects of the information technology: Hardware, Software, Communication, and Data. Data here is actually the input and the output of system, so it will not be considered as a part of it.

How can we represent these three aspects left, as functions in an operating system?

I'll consider the four main aspects of Information Technology as the main functional requirements in operating systems

In practical, these Functional requirements can be represented like this:

Hardware	Device Driver
Software	Software Applications
Communication	Protocols
Data	It's the input/output of the system, so it will not be considered as a part of the system, and thus, the system requirements

Device Driver

From this aspect, you might find Linux distributions that recognize plenty of devices, while others may not. But if you didn't find a built-in support for a device, it will be very difficult to add it, while it's much easier to do it in Windows.

So Windows here is better.

Software Applications

In general you can find the most common software categories in both windows and Linux (Internet browser, multimedia player, email client...etc).

But regarding the productivity software, or specialist software, such as video mixing, CAD, Windows is better.

Also there's a significant difference between Ubuntu and Slax, since Slax doesn't has a reliable package manger comparing to Ubuntu which will give a chance to find a lot of software packages.

Protocols

Linux supports IPv6 and SSH, while Windows isn't.

PERFORMANCE

Modularity

System usually is a set of components, or modules. As much as the number of modules is big, their size is small, system will be maintainable, but the relationship and the communication between each other will be bigger, which; in turn; reduce the system performance. And vice versa.

Regarding Linux, the separation between the kernel and the GUI engine (GTK+/Qt) makes Linux GUI much slower than Windows GUI. So, the GUI performance in Windows is better than the GUI performance in Linux.

Memory Consuming

It means how much system files consume from the hard disk and the RAM. Slax is the best here, Ubuntu will be the next, and the last one is Windows

Interoperability

It's the ability of the system to work in several platforms, which usually consist of hardware and an operating system. For example, I can say that *MyProg* has height interoperability if it was able to work in Intel-AMD/Windows, Intel-AMD/Linux, and Motorola/MacOS platforms.

But since our system here is an operating system, platform will be only the hardware.

In this case, we can consider Windows and Linux as equal, because both of them are typically designed for x386 architecture.

If we decided to talk interoperability on Software Application level, the judgment will be completely different.

The most common office suite in Linux, which is OpenOffice, is java based application. And since Java is natively designed to provide height interoperability, so that java-based software will work in any software platform. This means that OpenOffice, just like any other java-based application, will need a middleware (The virtual machine), that; in tern; will consume a significant amount of resources for not a real value-added functionality from user perspective, but maybe form the developer perspective, who might be interested in interoperability for his/her product. And that makes OpenOffice looks like a set of “blind components”, the refuses to take advantage of system-specific resources.

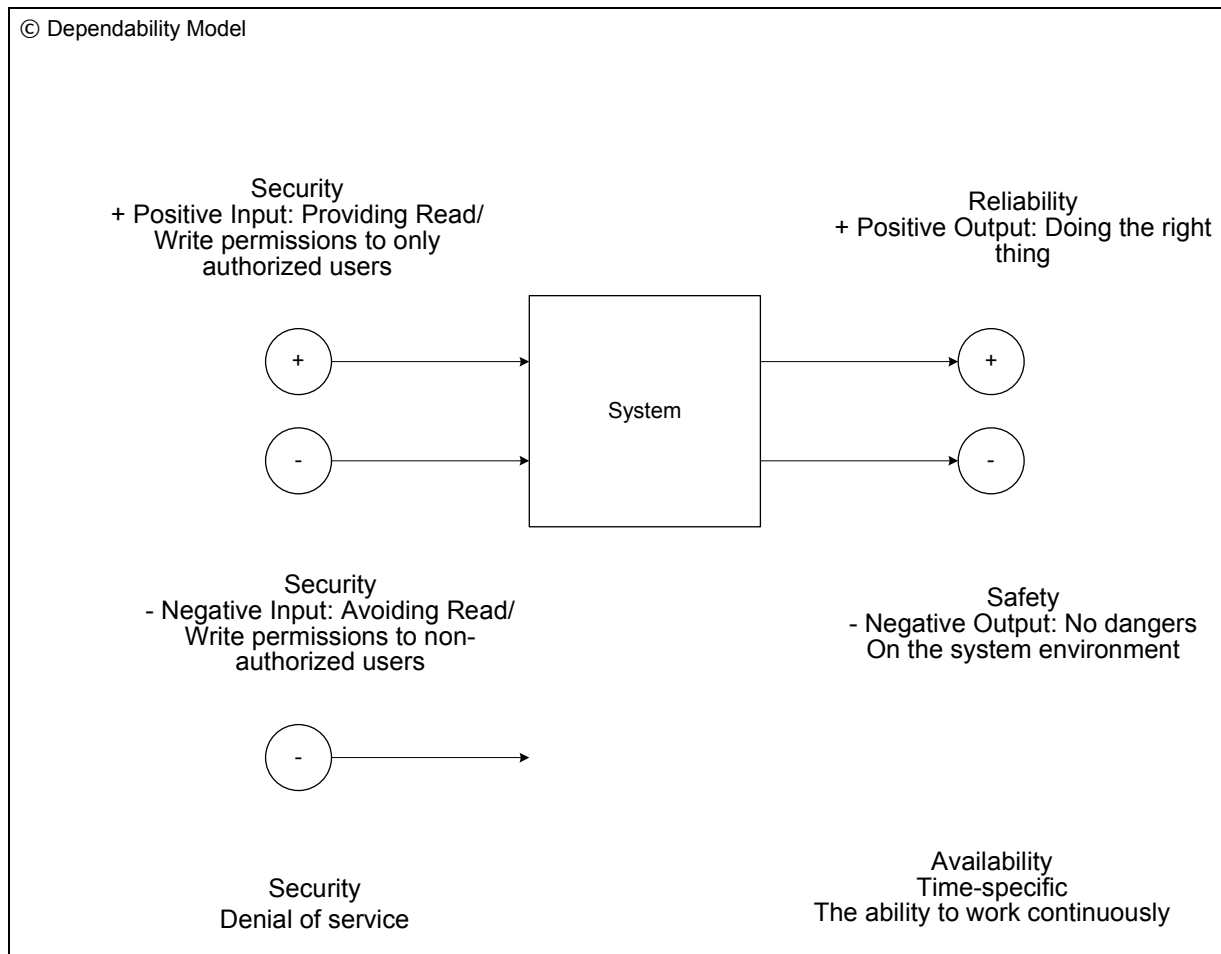
Since office applications are considered as the most important application in a desktop environment, we can conclude to the fact that Linux loses a significant amount of it total performance because of the dependency of one of the most important functional aspects on it, on a middleware.

That’s exactly what we can say about Ubuntu. By contrast, Slax is a pure-KDE based desktop environment, includes the KDE office suite which is called KOffice. And that makes it better in resource consuming. Windows will be the best here.

Notice that interoperability isn't a user requirement as much as it as a producer requirement. The party how will get benefit from this property is the producer, since his product will cover a wider market share, but user typically has a one platform, he/she will not care if this software-product will work in another platform or not. This, interoperability will not give user any value-added. Not only that but also it forces his/her system to perform a useless resource-consuming processes.

This shows and clarifies the age-old tension between user requirements and producer (commercial) requirements.

DEPENDABILITY



In a simple compression of the number of viruses, Trojans, error messages, and hang-up times, we can conclude here that dependability of Linux is far better than it in windows.

USABILITY

I will consider consistency of the (Look and Feel) of the desktop environment, and the Application-OperatingSystem interoperability as the most important factors for the usability.

Application-OperatingSystem interoperability is what I've talked about, when I was talking about the interoperability of OpenOffice. Using different GUI technology in one desktop can reduce usability very much.

According to first factor, Windows 2000/Me was the best, everything have the same spirit, while Linux has a very bad reputation in this aspect.

This due to the fact that the most distributions consists of several applications developed from different technologies (Java/GTK+/Qt), which there widgets have different appearance (attributes) and behaviors.

The only distribution that takes care of this factor and bring user to a one-spirit desktop environment, is Slax; a distribution based on pure KDE (K Desktop Environment), which in turn, has an ugly look and feel; a "Consistency of Ugliness"!

Windows and Slax will be equal.

PORTABILITY

This is one of the most significant features/requirements that keeps Windows behind Linux.

With LiveCD technology you can store a “working operating system” in a read-only medium, and boot from it to an access a ready and full-featured desktop environment.

According to this factor, Slax 5.0.6 and MEPIS is the best, sine it offers an “installable live operating system”.

Ubuntu comes next; it offers an Installable, and a live release of its operating system in two separated mediums.

While this factor conflicts with Microsoft's ‘organizational requirements’: the profitability, since portability makes it easy for users to share illegal copies of the system, which, in turn, could affect its revenue level.

Installability is a critical factor in portability.

Regarding the ease of installation, Slax 5.0.6 and MIPES, is the best, since these distributions have a GUI-based installer from a LiveCD, just like any application in the desktop.

Windows 2000/XP offers a very logical wizard from its installation CD.

Slax Installer fell by the wayside in the latest versions of Slax, and I think that this decision looked like reasonable, sine ease of critical functions, such as installation function, conflicts with safety requirement.

PRODUCER REQUIREMENTS

Implementation Requirements

One of the most significant example here is time constrain, which might force the producer to ignore some functions or requirements. That's exactly what happened with Microsoft when decided to ignore some features, like its new database-based file system in releasing the final release of Windows Vista.

Non-commercial software development doesn't have this problem, since there's no deadline.

Delivery Requirements

Getting the fees and protecting the copyright is the most important example fro delivery requirements. For example technically, Microsoft can develop a Live CD version of windows, but it will not do it, in order to prevent the ease redistribution of illegal copies of Windows. Linux doesn't have this limitation.

COMPERSSION TABLE

1= the best, 2= Good, 3= the worst

Requirements		Windows	Ubuntu	Slax	Hints
Functionality					
	Drivers	2	2	2	Availability/Ease of installation
	Software	1	2	3	Productivity Software
	Protocols	3	1	1	IP6/SSH
Performance					
	Modularity	1	2	2	GUI
	Memory Consuming (HD/RAM)	3	2	1	
	Interoperability	2	3	3	OpenOffice
Dependability					
	Reliability	3	1	1	bug messages
	Availability	3	1	1	hanging

	Security	3	1	1	Trojans (Privacy)
	Safety	3	1	1	Viruses
Usability		1	2	3	Consistency - Interoperability
Portability		3	2	1	LiveCD
Producer Requirements					
	Implementation	3	1	1	Deadline
	Delivery	3	1	1	Copyright

CONCLUSION

According to the software Requirement Network, you can see that the relationship between requirements, including the relationship between user requirements and producer requirements, is very complex; each can affect one or more of the rest, tensingⁱⁱ or supporting. And since the nature of the producer plays the significant role in determining the implementation and delivery requirements, which –in turn- can affect the user requirements, we can conclude that the difference in the nature of producer between these two systems can occur a significant difference in the whole of user requirements.

That's what we can say regarding the producer requirements.

Regarding to user requirements, Windows looks like an American car, a non-efficient comfortable car, while Linux looks like a complicated but powerful space ship!

So, since desktop users how are usually interested in usability more than anything else, desktop users are usually Windows users

While server market is usually interested in dependability more than anything else, Linux is the best for servers.

	Windows	Linux
Current status	For desktop users	For Server market
Should take care of	Dependability (especially Security, reliability) Interoperability	Usability Performance

REFERENCES

ⁱ DistroWatch.com: <http://distrowatch.com/dwres.php?resource=major>

ⁱⁱ Sommerville, I. (2007) Software Engineering, Eighth Edition, Addison-Wesley.