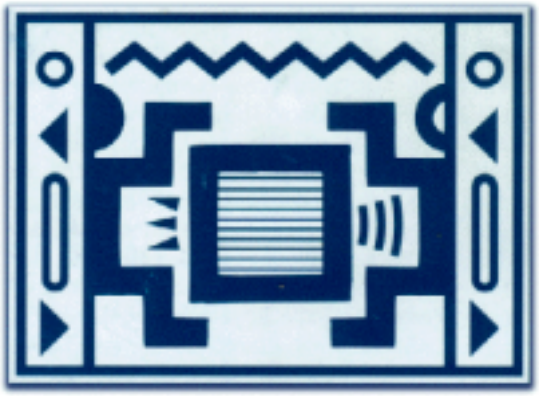


European Masters
in Interactive Multimedia Projects

Organisés à Bruxelles par l'Institut Supérieur de Commerce Saint-Louis
et l'Institut Supérieur de Formation Sociale et de Communication
avec le soutien du
Programme LEONARDO da VINCI de la Commission européenne

Open Technologies for an Open World

Open Standards, Open Source, Open Mind

Document version : 1.3

File: OpenTechnologies v13 L02.doc
Revision: 5/07/2003 - 10:12 PM

Table of Contents

Foreword

- **Motivations**

"Sometimes dreams are all that separate us from the machines" – Dan Simmons in The Fall of Hyperion

"Sometimes (…) the shortest route to courage is absolute ignorance" – Dan Simmons in Endymion

Recently we have heard a lot about Linux. Several discussions have been held in the professional circles, and the media is getting the message to the general public. It can be considered the major open source product, and the responsible for this public awareness. However, the concepts behind open source must also be explained, and - beyond this - the existence of open standards and protocols shall be understood. The comparison between "open" and proprietary software should not be limited to the cost, neither be influenced by the impression that open means free. What is the importance of this understanding for multimedia project managers?

There is a general theory about the transparency of the computer infrastructure for the Internet and intranet implementations, in phases like design, project management and decision-making. During this document, I aim to prove the importance of a general understanding of the hardware, software and communication infrastructure, to improve the overall quality of the project and the decision making process. The first battle between open and proprietary solutions will take place in the technological field. And this is only the beginning of the war.

The knowledge acquired when studying the impact of standards and open source for the Internet infrastructure, may be used to understand the consequences in other levels, like the design, the development and the exchange of structured information. Intensive research has been performed in the last decade, now the solutions are ready to be deployed. It's time to participate in the dialogs, discussion and analysis, which will make the standards for the future. It's essential to understand the social and economical rationales behind open and proprietary solutions, and how the political world can help to build a technological future.

This is an ambitious thought, I agree. Nevertheless, the ongoing social movements using the peer-to-peer concept are showing the power of tightly coupled motivated persons. Linux is there to prove it.

- **Sources**

I tried to be eclectic, using some books and syllabi as basic references for the technological, economical and social analysis, my 2-year collection of Datanews magazines to analyse the current evolutions of the Belgian and European market and political environment, seminars and the Internet to complete the evaluation from a global and updated standpoint. These are always fully referenced in the footnotes, with complete details being given in the Appendix D.

- **Method**

There are many different technological aspects discussed in this document. At a first sight, it may look like a collection of superficial investigations about different topics. However, it aims to discuss open solutions in a very broad scope. So I studied every information technology aspect, which is at the same time important to the development of multimedia applications and currently with interesting questions about the choice of open or proprietary alternatives. A brief explanation about the technology itself is always given in the beginning of each chapter and paragraph, to allow the reading of this document without pre-requisites. The focus of the discussion is always around open source and open standards. To allow this document to be read in modules or in non-sequential browsing, the conclusions are often done in the end of each chapter and paragraphs.

- **The light side of the moon**

There are two ways of reading this thesis. The shortest and funny way is to go directly to the online conclusion (http://www.k-binder.be/INCA/), play around with the suggested architecture for the future, and then come back to the document and read only the topics that really interest and attract you. Then disagree with my standpoint and tell me why (by using the website to exchange opinions, the readers can have an open discussion around the topics discussed in this document, which can be a basis for the effective design of the new architecture).

The second way is to continue reading this document.

- **Acknowledgements**

Many thanks to my wife Joyce for the strong support during the long research evenings, nights and weekends. Big thanks to my sister Mariana who helped me to keep concentrated to finish this work. The open spirit of my daughter and family, from the cradle, were also fundamental for my motivations.

Special regards to the Professor Jean-Luc Vidick for the help to structure this document in its final form, and to complete it with important topics about the extreme development; to the professor Michel Bauwens for the interesting discussions that helped me to find the target subjects and to compose the bibliography; to the professor Attila Darabos for the final revision; and to the professor Pierre Rummens for the remarks about the research methods and the document formats and structures.

I hope you enjoy the reading, and if you feel motivated by my ideas please feel free to help me improving this document[1].

[1] mailto:jean@k-binder.be

1. Introduction

1.1. Structure

"Let's say there are four steps. Four stages. Four levels. The first Is learning the language of the dead, the second is to learn the language of the living, the third is hearing the music of the spheres. The fourth step is learning to take the first step." – Dan Simmons in The Fall of Hyperion

The standardization is an important point to be considered in the evolution of the different technologies, and so is their openness. It gives developers the possibility to understand the infrastructure requirements, to create compatible products, to participate in their evolution, to innovate. This openness may be well represented in the source code, the "alma mater" of every computer program. We need to understand "Open Source" as the concept created by computer programmers united under the hackers ethic. However, we should not be limited by it: We shall go one abstraction level further to define the "Open" concept, by analysing also the Open Systems, the Open Standards and the Open Platforms. The first part of the analysis will be divided into three different tiers, which compose together the realm of IT and Internet today. They are the infrastructure, the development and the information exchange (see figure below).

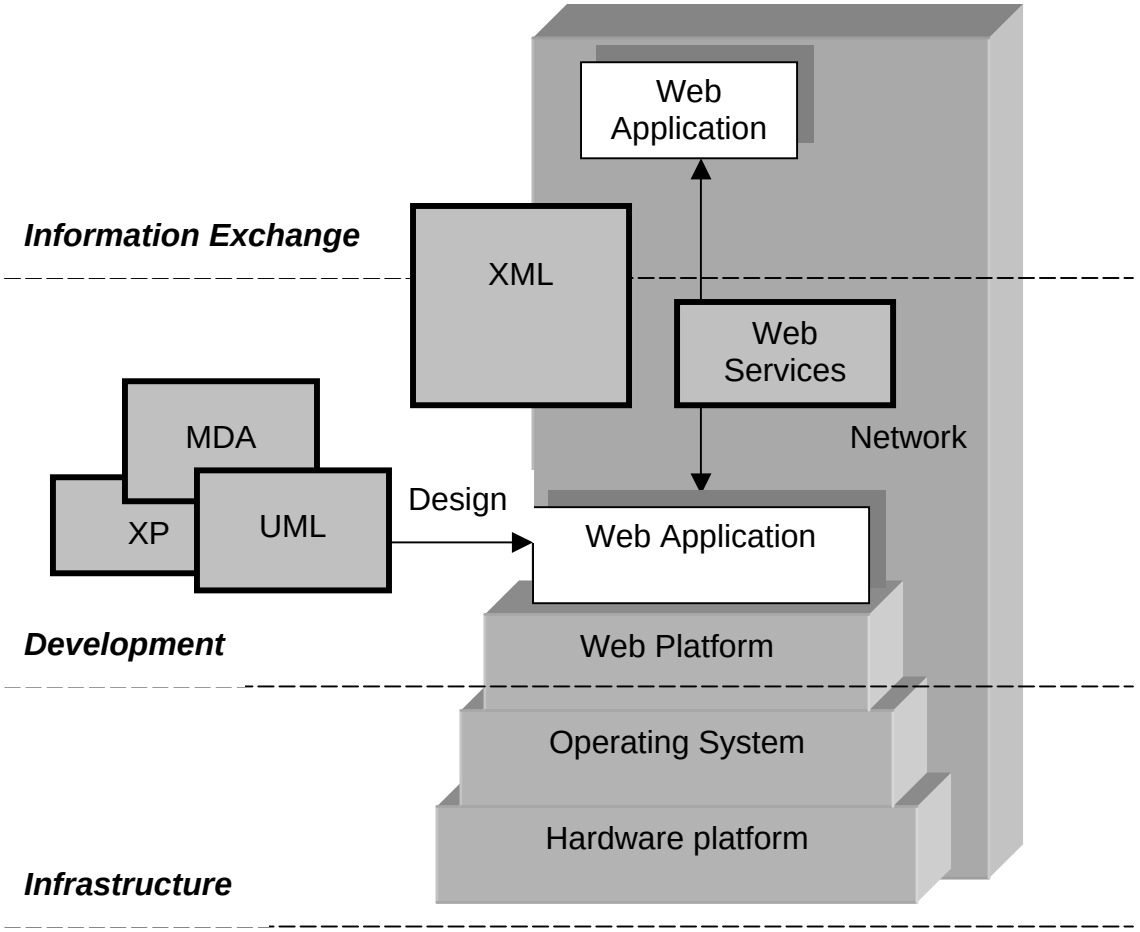


Figure 1 – The three tiers under the scope of the first part of this analysis

Let's start by the bottom-level tier, and discuss in the first chapter the hardware and the operating systems used to build the Internet infrastructure. Different alternatives are investigated, trying to understand their original goals, their evolution, their current situation and the tendencies. The analysis is always focused on the applicability of the platforms as servers taking part on the Internet infrastructure. And what is the base of Internet? The standards. They participated in the spread of the Net from the very beginning, so they do in this document. The goal is to understand what are the different hardware platforms, the software environments, their concepts, history and targets, and how they became standards. We will also analyse what benefits we may expect from the open environments, together with the level of openness available from the common proprietary alternatives. We conclude with the role of standards on the creation of network protocols used worldwide, tantamount for the success of Internet.

In the second tier - development - we can consider the ongoing battle for a common web platform and the role of the web services. This is excellent for a case study, as we have three different warriors: An open source, an open platform, and a proprietary solution. Additionally we will analyse the usage of common rules to exchange information about the design of applications and data, with the goal of technology independence: MDA and UML. We briefly discuss the extreme programming concepts and their relation with the open source products and projects.

The analysis ends with the third tier – information exchange - by exploiting the benefits of a common language to exchange information and data definitions, coupled and structured. This is done by XML and its derivates.

The aim of the second part is to analyse the technologies by a broader standpoint. We should be able to trace some conclusions and imagine future trends and proposals. Using the cases detailed in the first part, we will analyse some of the basic concepts around the network society and the new economy, trying to understand the impact of standards in the usage of technological power, the strategies currently used by the enterprises, the comparative elements between open and proprietary solutions. Finally, we will discuss the role of politics, to enforce or stimulate the usage of open technology.

This study contains background material that introduces some important topics for readers who are not familiar with them. References are provided for those who want more complete understanding. The hyperlinks fans should refer to the Appendix A, on page 127, and feel free to use the online version on www.k-binder.be/Papers/, which intend to be often up-to-date. As an *avant-goût*, the basic concepts used during the rest of the document.

1.2. Open

To define the *Open* general concept, under the scope of this study, let us consider two definitions, with the help from the dictionary[1]:

Open

1: having no enclosing or confining barrier: accessible on all or nearly all sides
5: not restricted to a particular group or category of participants

Based on this, we can consider as "*Open"* any product, concept, idea or standard:

- That is freely available to be researched, investigated, analysed and used with respect to the intellectual property;
- That can be adapted, completed and updated by any person or company interested in improving its quality or the functionality, possibly coordinated by a person or organization.

Let us now analyse the open model, used for Open source development and the establishment of open standards.

1.3. The Open Model

1.3.1. Open Source

- **Definition**

Historically, the makers of proprietary software have generally not made source code available. Defined as "any program whose source code is made available for use or modification as users or other developers see fit" [2], open source software is usually developed as a public collaboration and made freely available.

Open Source is a certification mark owned by the Open Source Initiative (OSI). Developers of open software (software intended to be freely shared, potentially improved and redistributed by others) can use the Open Source trademark if their distribution terms conform to the OSI's Open Source Definition[3]. All the terms below must be applied together, for the product and its license, and in all cases:

- Free Redistribution – The software being distributed must be redistributed to anyone else without any restriction.
- Source Code – The source code must be made available (so that the receiving party will be able to improve or modify it)

[1] Source: Webster's Revised Unabridged Dictionary, © 1996, 1998 MICRA, Inc.
[2] Source: searchSolaris.com
[3] Source: www.opensource.org/osd.html

- Derived works – The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

- Integrity of the Author's Source Code – The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

- No discrimination against persons or groups

- No discrimination against fields of endeavour – It may not restrict the program from being used in a business, or from being used for a specific type of research.

- Distribution of license – The rights attached to the program must apply to whom the program is redistributed without the need for execution of an additional license by those parties. In other words, the license must be automatic, no signature required.

- License must not be specific to a product – A product identified as Open Source cannot be free only if used in a particular brand of Linux distribution.

- License must not contaminate other software – The license must not place restrictions on other software that is distributed along with the licensed software.

As Perens summarizes, from the programmer's standpoint, these are the rights when using Open Source programs:

- The right to make copies of the program, and distribute those copies.

- The right to have access to the software's source code, a necessary preliminary before you can change it

- The right to make improvements to the program[4]

- **Licenses**

There are many different types of licenses used by Open Source products. The most common are[5]:

- Public Domain – A public-domain program is one upon which the author has deliberately surrendered his copyright rights. It can't really be said to come with a license. A public domain program can even be re-licensed, its version removed from public domain, with the author name being replaced by any other name.

[4] Source: bibliography 5 – page 172.
[5] Source: bibliography 5 – page 182.

- GPL (GNU General Public License) – the GPL is a political manifesto as well as a software license, and much of its text is concerned with explaining the rationale behind the license. GPL satisfies the Open Source Definition. However, GPL does not guarantee the integrity of the author's source code, forces the modifications to be distributed under the GPL and does not allow the incorporation of a GPL program into a proprietary program.

- LGPL (GNU Library General Public License) – The LGPL is a derivative of the GPL that was designed for software libraries. Unlike the GPL, a LGPL program can be incorporated into a proprietary program.

- X, BSD and Apache – These licenses let you do nearly anything with the software licensed under them. This is because their origin was to cover software funded by monetary grants of the US government.

- NPL (Netscape Public License) – This license has been prepared to give Netscape the privilege of re-licensing modifications made to their software. They can take those modifications private, improve them, and refuse to give the result to anybody (including the authors of the modifications).

- MPL (Mozilla Public License) – The MPL is similar to the NPL, but does not contain the clause that allows Netscape to re-license the modifications.

License	Can be mixed with non-free software	Modifications can be taken private and not returned to their authors	Can be re-licensed by anyone	Contains special privileges for the original copyright holder over the modifications
Public Domain	√	√	√	
GPL				
LGPL	√			
BSD	√	√		
NPL	√	√		√
MPL	√	√		

Table 1 – Comparison of Open Source licensing practices[6]

- **Open Source software engineering**

The traditional software engineering process generally consists of marketing requirements, system-level design, detailed design, implementation, integration, field-testing, documentation and support. An Open Source project can include every single one of these elements:

- The marketing requirements are normally discussed by using a mailing list or newsgroup, where the needs from one member of the community are reviewed and complemented by the peers. Failure to obtain consensus results in "code splits", where other developers start releasing their own versions.

[6] Source: Bibliography 5, page 185

- There is usually no system-level design for a hacker-initiated Open Source development. A basic design allows the first release of code to be built, and then revisions are made by the community. After some versions, the system design is implicitly defined, and sometimes it is written down in the documentation.

- Detailed design is normally absent of pure open source initiatives, mostly because most of the community is able to read the code directly, interpret the routines, functions and parameters, and modifying them as required. This makes further development more difficult and time-consuming.

- Implementation is the primary motivation for almost all Open Source software development effort ever expended. It is how most programmers experiment with new styles, ideas and techniques.

- Integration usually involves organizing the programs, libraries and instructions in such a way they can be used by users in other systems and equipments to effectively use the software.

- Field-testing is one of the major strengths of Open Source development. When the marketing phase has been effective, many potential users are waiting for the first versions to be available, and willing to install them, test and make suggestions and correct bugs.

- The documentation is usually written in a very informal language, free style and usually funny way. Often websites are created to allow the documentation to be provided and completed by the user community, and becomes a potential source for examples, "tips and tricks". One of the brightest examples is the online documentation for PHP[7].

- The support is normally provided via FAQS and discussion lists or even by the developers themselves by e-mail, in a "best effort" basis, depending on their availability, willingness and ability to answer the question or correct the problem. The lack of official support can keep some users (and many companies) away from Open Source programs, but it also creates opportunities for consultants or software distributors to sell support contracts and/or enhanced commercial versions.

The commercial versions of Open Source software (like BSD, BIND and Sendmail) often use the original Open Source code, developed using the hackers' model, later refined by most of the phases described above.

- **Open-source cycle**

Many Open Source software start with an idea, discussed via the Internet, developed by the community, implemented as first draft versions, and consolidated into final versions after a lot of debugging. After this final software starts to be used globally,

[7] www.php.net

and interesting companies, the authors may decide to start receiving some financial compensation for their hard work.

Then an organization may be created, or some existing company can start to distribute the product alone or bundled with other similar pieces of software. Sometimes, after a reasonable funding is raised and more development is done (now remunerated) a commercial version of the product may be released, often with more functionality than the free version, and sometimes without the source code being generally distributed. This is attributed to the difficulty of small companies to remain profitable by distributing only open source software.

- **Open Source Science**

As argued by DiBona, Ockman and Stone[8], "Science is ultimately an Open Source enterprise. The scientific method rests on a process of discovery, and a process of justification. For scientific results to be justified, they must be replicable. Replication is not possible unless the source is shared: the hypothesis, the test conditions, and the results. The process of discovery can follow many paths, and at times scientific discoveries do occur in isolation. But ultimately the process of discovery must be served by sharing information: enabling other scientists to go forward where one cannot; pollinating the ideas of others so that something new may grow that otherwise would not have been born."

Ultimately, the Open Source movement is an extension of the scientific method, because at the heart of the computer industry is computer science. Computer science differs from all other sciences, by having one means of enabling peers to replicate results: share the source code. To demonstrate the validity of a program, the means to compile and run the program must be provided.

Himanen considers that the scientists have developed this method "not only for ethical reasons but also because it has proved to be the most successful way of creating scientific knowledge. All of our understanding of nature is based on this academic or scientific model. The reason why the original hackers' Open Source model works so effectively seems to be – in addition to the facts that they are realizing their passions and are motivated by peer recognition, as scientists are also – that to a great degree it confirms to the ideal open academic model, which is historically the best adapted for information creation"[9].

1.3.2. The Open Standards

One of the important factors in the success of the Internet comes from the "governance mechanisms" (rather than regulation) that guide its use and evolution, in particular, the direct focus on inter-connection and interoperability among the various constituent networks.

[8] See bibliography 5 – Introduction.
[9] See bibliography 4, page 69

- **Standards**

The standards are fundamental for the network economy. They allow the companies to be connected by establishing clear communication rules and protocols. For manufacturing networks, composed by the company with the product design, the suppliers of different components and the assembly lines, standards guarantee the compatibility of all the different parts in the production process. In the information and technology networks, the standards guarantee the compatibility of the infrastructure components analysed in the first chapter – hardware, operating systems and application software – and the interoperability of the companies via the Internet with clear network protocols.

Standardization is by definition a political, economical and technological process aiming to establish a set of rules. These are documented agreements containing technical specifications or other precise criteria to be used as rules, directions or definitions. Thus, equipments, products, processes and services based in the same set of rules are fully compatible with each other.

- **De facto or De jure**

Standards are normally classified according to their nature. De facto standards are products and protocols which conquer the market by establishing a network of interconnected and compatible products, and gaining recognition from the consumers. An example is the set of de facto standards built around the IBM PC specification.

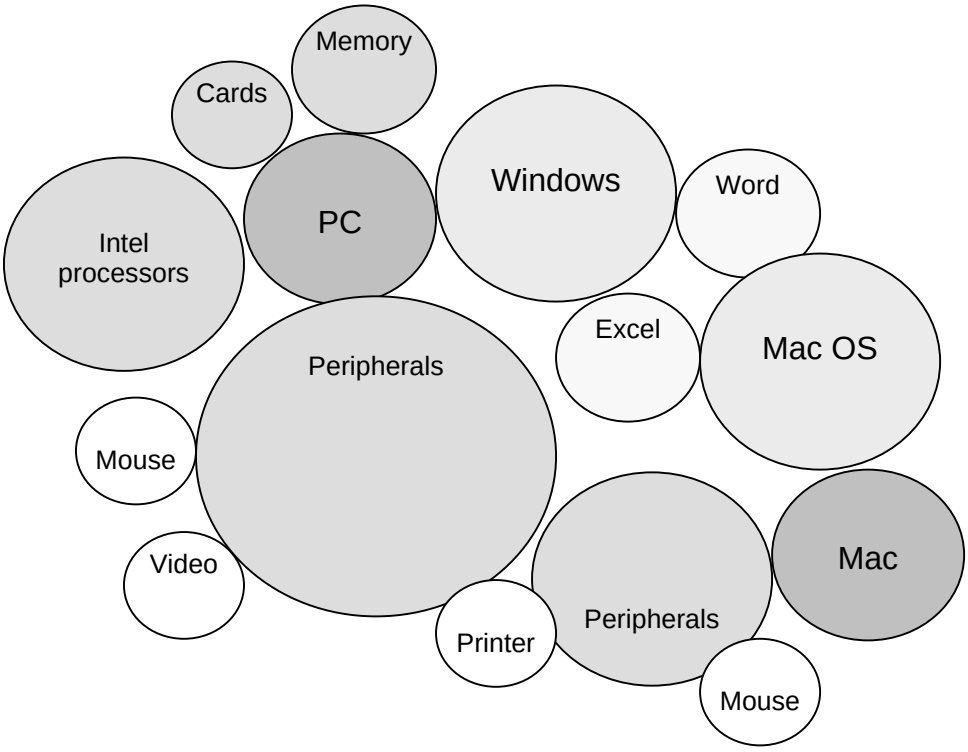


Figure 2 – Examples of de facto standards and their connectivity

De facto standards are normally preferred by companies because they don't need to follow a complex and time-lengthy process for its validation. Often – as the case of

Windows – they are accepted by the market even without being elaborated by scientific process, and without strong R&D investments. These are also the reasons by which they are not easily accepted by normalisation organisations, by the scientific institutions and by the academic circle.

By opposition, de jure standards are often based on a new technology, created by a company that accept to publish their concepts and definitions for a review of the academic and scientific communities, and be officially approved by a normalisation institution. A good example is the OSI standard and the protocols compatible with its different tiers (see the Figure 14 on page 42).

Some of the main motivations for this are the ability to compete with existing dominant standards, to easily found a cooperation network of software and hardware suppliers, to obtain credibility from the academic world and consequently gathering the cooperation from the students community.

A good practice from the really independent and non-profit normalisation organisations is to only accept to recognise open standards. This is necessary to avoid the formation of monopolies and unfair commercial practices by the royalty-owners. This has recently been contested when standards for web services have been analysed.

De facto standards	***De jure standards***
Strengths: - Quickly developed - Optimal solution for the original goal	**Weaknesses**: - Lengthy development - Solution can be used beyond the original goal - Compromise of technical choices
Weaknesses: - Short-sighted - Not clearly defined - Stimulates monopolies (by copyrights) - Multiple solutions for the same type of applications	**Strengths**: - Long term analysis by experts - Clear and complete definitions - Stimulates competition - Standard controlled by independent organizations rather than companies - Possibility of independent certifications

Table 2 – Comparison between de facto and de juri standards[10]

- **Mare liberum or Mare clausum**

One standard is known as proprietary when it has been developed by a company, which remains the owner of royalties that limit the usage of the standard specifications by the payment of licence fees.

The Open Standards specifications are open to the public and can be freely implemented by any developer. Open standards are usually developed and maintained by formal bodies and/or communities of interested parties, such as the Free Software/Open Source community. Open standards exist in opposition to the proprietary standards and work to ensure that the widest possible group of

[10] Source: Bibliography 27, page 5

contemporary readers may access a publication. In a world of multiple hardware and software platforms, it is virtually impossible to guarantee that a given electronic publication will retain its intended look and feel for all viewers, but open standards at least increase the likelihood that a publication can be opened in some form.

From a business perspective, open standards help to ensure that product development and debugging occurs quickly, cheaply and effectively by dispersing these tasks among wide groups of users. Open standards also work to promote customer loyalty, because the use of open standards suggests that a company trusts its clients and is willing to engage in honest conversations with them. Criteria for open standard products include: absence of specificity to a particular vendor, wide distribution of standards, and easy and free or low-cost accessibility.

- **Protocols**

A protocol is the special set of rules used by different network layers, allowing them to exchange information and work cohesively. In a point-to-point connection, there are protocols between each of the several layers and each corresponding layer at the other end of a communication.[11]

The TCP/IP protocols (TCP, IP, HTTP…) are now quite old and probably much less efficient than newer approaches to high-speed date networking (e.g. Frame Relay or SMDS). Their success stems from the fact that the Internet is often today the only possible outlet that offers a standardized and stable interface, along with a deliberate focus on openness and interconnection. This makes the Internet extremely attractive for very different groups of users, ranging from corporations to academic institutions. By contrast with traditional telecommunications networks, the rules governing the Internet focus on interconnection and interoperability, rather than attempting to closely define the types of applications that are allowed or the rate of return permitted to its constituents.[12]

- **Architectures or Platforms**

Architecture is a term applied to both the process and the outcome of thinking out and specifying the overall structure, logical components, and the logical interrelationships of a computer, its operating system, a network, or other conception. Architectures can be specific (e.g. IBM 360, Intel Pentium) or reference models (e.g. OSI). Computer architectures can be divided into five fundamental components (or subsystems): processing, control, storage, input/output, and communication. Each of them may have a design that has to fit into the overall architecture, and sometimes constitute an independent architecture[13].

As it will be exploited in the chapter 2.2.1 (*z/VM and z/OS*), the usage of open architectures – even when maintained by commercial companies - can bring many technical advantages to the suppliers of the building blocks (Software, hardware and network components) and mainly to the user community, which can profit from a large network of compatible components, ensuring an independence and favouring

[11] Source: www.whatis.com
[12] Source: bibliography 62
[13] Source: www.whatis.com

concurrence, creativity and innovation. The economic potential will be analysed in details on the chapter 6.3 (*Feedback*).

The term platform may be used as a synonym for architecture, and sometimes may be used to designate their practical implementations.

- **Certification**

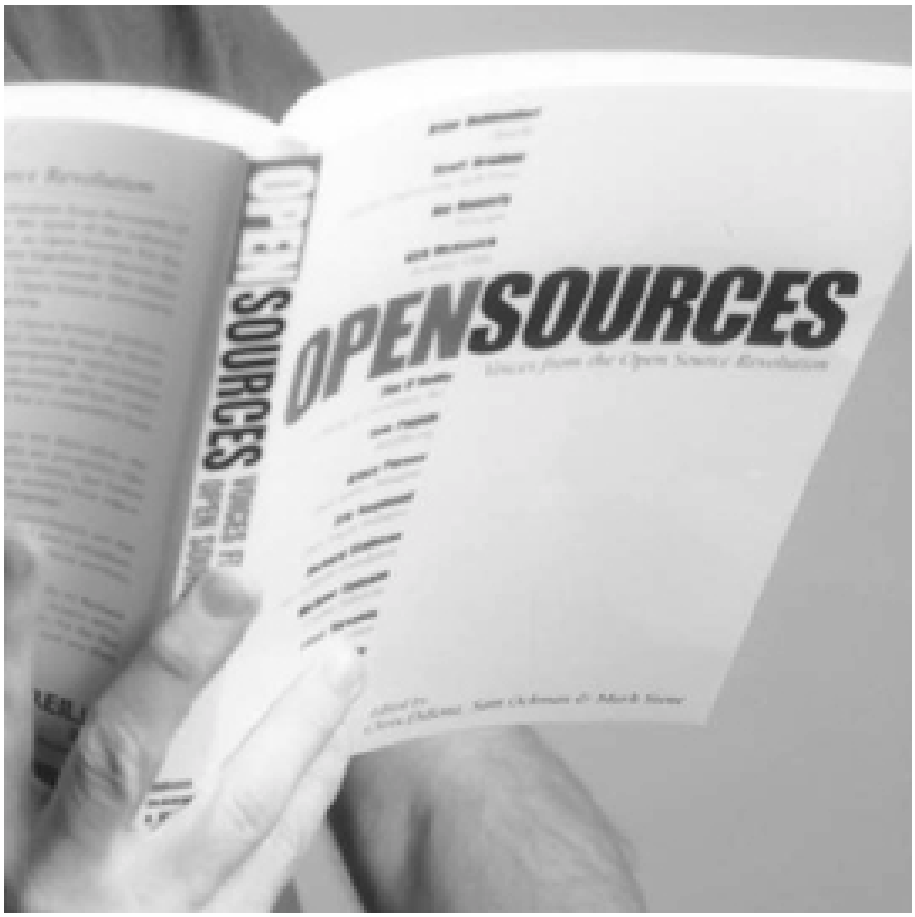
Certify is to validate that products, materials, services, systems or persons are compliant to the standard specifications. One chapter of the standard definitions is always dedicated to the certification steps to be performed, and the specifications to be verified.

The certification process may be performed by any entity. However, it's normally performed by independent organizations as it may be expensive and requires specialized technological knowledge.[14]

[14] Source: Bibliography 27, page 6

Part I – Open Technologies

"It's the question that drives us. The question that brought you here. You know the question, just as I did." Trinity in *The Matrix*



2. Open Infrastructure

"Technology is neither good nor bad, nor is it neutral" – Melvin Kranzberg

According to Patrick Gerland, "the three essentials of computer technology are: hardware - the physical machinery that stores and processes information -, software - the programs and procedures that orchestrate and control the operation of hardware - and people who know what the hardware and software can do and who accordingly design and implement applications that exert appropriate commands and controls over hardware and software to achieve desired results".[15]

Despite the broad discussion about open source and its influence on the evolution of hardware and software, there's a general lack of knowledge about the different types of operating systems, their original and current targets, and their relationship with the hardware platforms and architectures. For a better understanding of the further topics, let's start by discussing this infrastructure and analyse the status of the hardware and basic software. This analysis will also solidify the definition of architecture, standards and protocols, to be used on the following chapters. We will see, for example, that new concepts as Open Source and ASP are largely inspired in their parents Open Systems and Time Sharing.

The hardware and software components are sometimes generically referred to as middleware.

2.1. Hardware

Hardware is a broad term. It covers all the equipment used to process information electronically, and it's traditionally divided into two main categories: computers and peripherals[16]. In this section, we will concentrate our analysis in the evolution of the computers, from owned machines into commodities. Peripheral references to the peripherals will be done. Let us consider the division of the computer into servers and clients, discuss the traditional server families, and the new concept of servers. After, we will briefly analyse the machines in the client-side.

2.1.1. Traditional Server Families

In the beginning, they were simply called computers. And they were large-scale computers. In the mid-1980's, after the surge of the personal computers, we started to classify the computers according to their capacity, their price, their "original" goal and their target users. This classification remained until the end of the 1990s, and the three different families will now be briefly described.

[15] See bibliography 26
[16] For a more complete description, see http://www.wikipedia.org/wiki/Computer.

- **Mainframes**

Let's not talk about the big computers filling entire rooms, weighting tons and composed by valves[17]. This is not a mainframe any more, for a long time. Let's think about relatively small servers (they are often not larger than a fridge) with big power capacity, enough to run online applications for thousands of users spread over different countries, and to process batch jobs which can process millions of nightly database updates. They have never been sexy (and this was not the goal: we are talking about a period between 1970 and 1985, when graphics, colours, sound and animation were not part of the requirements for a professional computer application) and today it seems that the only alternative for them to survive is to mimic and co-operate with open systems environments.

Usually mainframe applications are referred to as "legacy" ("Something received from the past[18]"). One can argue the usage of this term[19], mainly because of the undeniable stability and performance of these machines and considering that most of the core applications for big companies are still processed using mainframe computers, connected with computers from other platforms to get profit from a better user interface.

Figure 3 – IBM ZSeries 900

Current mainframe brands are IBM zSeries, Hitachi M-Series, Fujitsu-Siemens PrimePower 2000 and BS2000 and Ahmdal Millenium[20]. The term mainframe is also being marketed by some vendors to designate high-end servers, which are a direct evolution from the mid-range servers, with a processing capacity (and price) aiming to compete with the traditional mainframes.

- **Mid-range servers**

This is a difficult assumption to make. The capacity of the machines in this category is increasing fast, and they are already claiming to be high-end servers, and winning market share from the mainframes. Let's consider it a separate category, for the moment.

Once upon a time, there were the minicomputers[21]. They used to fill the mid-range area between micros and mainframes. As of 2001, the term minicomputer is no longer used for the mid-range computer systems, and most are now referred to simply as servers[22]. Mid-range servers are machines able to perform complex

[17] To understand the computer history from the real beginning, please refer to http://www.sysprog.net/history.html. To see pictures from some old mainframes see http://www.piercefuller.com/collect/main.html.
[18] Source: Webster's Revised Unabridged Dictionary, © 1996, 1998 MICRA, Inc.
[19] The group bit.listserv.ibm-main is the mainframe discussion list by excellence. When looking for the term legacy (e.g.: http://groups.google.be/groups?q=legacy+group:bit.listserv.ibm-main) we may find more than a thousand posts.
[20] Several manufacturers produced mainframe computers in the 1960s and 1970s: Burroughs, Control Data, General Electric, Honeywell, NCR, RCA, and Univac – source: wikipedia.
[21] Some examples in http://www.piercefuller.com/collect/mini.html. A complete description may be found in http://www.wikipedia.org/wiki/Minicomputer.
[22] A complete description may be found in http://www.wikipedia.org/wiki/Minicomputer.

processing, with distributed databases, multi-processing and multi-tasking, and deliver service to the same number of users than a high-end server, but with lower cost and often with an inferior level of protection, stability and security.

The main suppliers in this area are HP and Sun.

Figure 4 – HP RISC rp8400

- **Low-range servers**

Initially the microcomputers were used as desktop, single-user machines. Then they started to assume functions from the minicomputers, like multi-tasking and multi-processing. Nowadays they can provide services to several users, host web applications, function as e-mail and database servers. Initially each server was used for a different function, and they were connected via the network. This is called multi-tier server architecture. With the evolution of the processing speed, today each server may be used for several functions.

They may also work as metaframes, providing service to users connected via client or network computers. All the applications run in the server side (that is required to have a very good processing capacity), which exchange the screens with the clients (that can be relatively slow).

There are several suppliers in this area, like Dell and Compaq.

Figure 5 – DELL Poweredge 1600SC

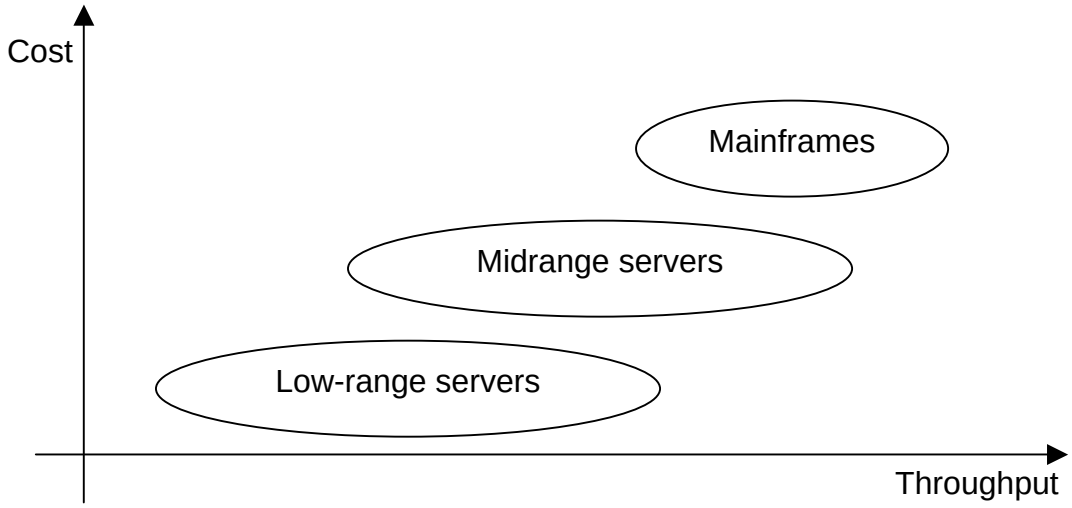


Figure 6 – Traditional Server families[23]

[23] The throughput represents the capacity of one determined environment to process several tr ansactions, or to serve several users, by keeping the same performance level.

2.1.2. Servers – The new generation

The classification categories discussed before are becoming more difficult to define and the division lines are blurring. The new generation of servers span a large range of price and processing capacity, starting by low-entry models, until powerful high-end servers, which can be tightly connected into clusters. This methodology is implemented – in different ways – by the most part of the hardware suppliers in the server market.

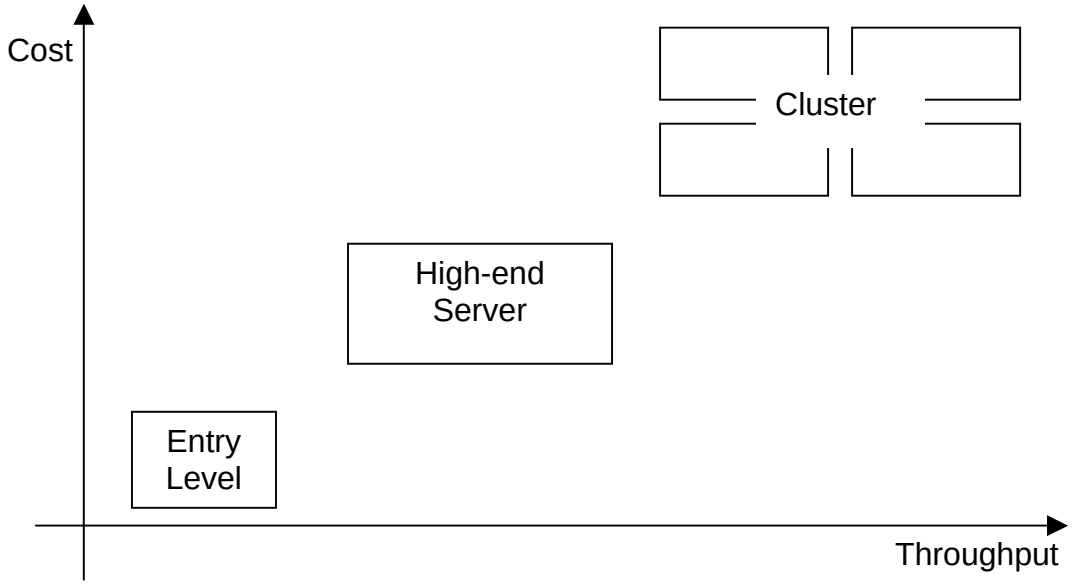


Figure 7 – The new generation of servers

The differences between the entry-level and high-end servers depend on the architecture and the supplier. They may imply the number of parallel processors, the type and capacity of each processor, the memory available, cache, and connections with the peripherals. The clusters are normally used for applications demanding intensive calculations, and are composed by several servers connected by the usage of high-speed channels (when they are situated in the same space) or high-speed network links (when the servers are geographically dispersed to provide high availability in cases of natural disasters). There are many different ways to implement the clusters to obtain a single point of control and maintenance, treating the cluster almost as a single server – without creating a single point of failure, which could imply the complete unavailability of the cluster in case of problems with one of the elements.

Estimate the throughput for future applications is only possible with a good capacity planning, prepared by experimented specialists. To find a server – as performance tests with the real charge are quite difficult to be elaborated - there are several benchmarks, organized by recognized independent companies, each one with a clear advantage for a different supplier. As the cost is extremely important - and when the hardware supplier participates in the capacity planning – a Service Level Agreement (SLA) may specify guarantees that the capacity installed will satisfy the estimated demand.

However, another differentiating factor, which is gaining importance nowadays, is the openness of the hardware and operating systems. This will be largely exploited further in this document.

2.1.3. Autonomic grid on demand

The current challenge, in the quest for the holy server, is to build flexible, scalable and resilient infrastructures, able to respond to unexpected surges in traffic and use. The solution may be found by implementing concepts like grid computing and autonomic computing, which are highly dependent on open standards and tightly related to open source.

- **Grid Computing**

Grid computing[24] is a new, service-oriented architecture that embraces heterogeneous systems and involves yoking together many cheap low-power computers - dispersed geographically - via open standards, to create a system with the high processing power typical of a large supercomputer, at a fraction of the price. The goal is to link – mainly through the Internet – computers and peripherals, by cumulating their processing and stocking capacities. All the connected systems may then profit from the set of assembled resources – like processing capacity, memory, disk, tapes, software and data – that are global and virtual at the same time.[25]

The challenges are to connect different machines and standards, to develop software able to manage and distribute the resources over the network, to create a development platform enabling programs to make profit of the parallel and distributed tasks, security issues (authentication, authorisation and policies) and to develop reliable and fast networks.

The basic principle is not new, as the HPC (high performance computing) technology is already used in academic and research settings, and the peer-to-peer has already been used to create a large network of personal computers (e.g. SETI@home). The target now is to expand this technology to the business area via associations like the Global Grid Forum (assembling more than 200 universities, laboratories and private companies).

Another initiative is the Globus project, formed by many American research institutes, responsible for Globus toolkit, which uses Open Source concepts for the basic software development, and has created the Open Grid Services Architecture (OGSA), a set of open and published specifications and standards for grid computing (including SOAP, WSDL and XML). The OGSA version 1.0 has been approved in December 2002. Other similar projects are Utility Computing (HP) and N1 (Sun)[26].

[24] For more information on grid computing: http://zdnet.com.com/2100-1105-863783.html (Globus protocols)
[25] Source: bibliography 36
[26] Source: Datanews n°03, 24/01/2003. More references may be found in the Web:
www.gridforum.org
www.globus.org
www.gridcomputing.com
www.gridpartners.com

With the costs to acquire, deploy, and maintain an expanding number of servers weighing down potential productivity gains, the market is shifting toward various concepts of service-centric computing. This may hold deep potential to not only lower the capital and operational costs of a data centre, but also to impart that infrastructure with the increased availability and agility to respond to an ever-changing business environment.

- **Autonomic Computing**[27]

As IT systems become more complex and difficult to maintain, alternative technologies must manage and improve its own operation with minimal human intervention. Autonomic computing is focused in making software and hardware that are self-optimising[28], self-configuring[29], self-protecting[30] and self-healing[31]. It is similar to the grid concepts, by embracing the development of intelligent, open systems that are capable of adapting to varying circumstances and preparing resources to efficiently handle the workloads placed upon them. Autonomic computers aren't a separate category of products. Rather, such capabilities are appearing in a growing number of systems.

The objective is to help companies more easily and cost-effectively manage their grid computing systems. It's expected that when autonomic computing reaches its full potential, information systems will run themselves based on set business policies and objectives. The implementation of such a system start with a trial phase, in which the system suggests actions and then wait for approval. After a fine-tuning of the rules, the system may run unattended.

It's believed that only a holistic, standards-based approach can achieve the full benefits of autonomic computing. Several standards bodies – including the Internet Engineering Task Force, Distributed Management Task Force and Global Grid Forum – are working together with private companies to leverage existing standards and develop new standards where none exist. Existing and emerging standards relevant to autonomic computing include:

- Common Information Model
- Policy, Simple Network Management Protocol (IETF)
- Organisation for the advancement of structured information standards
- Java management extensions
- Web Services Security

---

www.globusworld.org
www.gridtoday.com

[27] The term "Autonomic" comes from the autonomic nervous system, which controls many organs and muscles in the human body.

[28] Self-optimising refers to the ability of the IT environment to efficiently maximize resource allocation and utilization to meet end users' needs with minimal human intervention.

[29] With the ability to dynamically configure itself on the fly, an IT environment can adapt immediately – and with minimal human intervention – to the deployment of new components or changes in the IT environment.

[30] A self-protecting environment can detect hostile or intrusive behaviour as it occurs and take autonomous actions to make itself less vulnerable to unauthorized access and use, viruses, denial-of-service attacks and general failures.

[31] Self-healing environments can detect improper operations and then initiate corrective action without disrupting system applications.

One example of current technologies using autonomic components is disk servers with predictive failure analysis and pre-emptive RAID reconstructs, which are designed to monitor the system health and to detect potential problems or systems errors before harming the data. Performance optimisation is also obtained via intelligent cache management and I/O prioritisation[32].

2.1.4. Clients

- **Desktops (and laptops)**

In the beginning of the personal computers, hobby was the main objective, and there were several different platforms (Amiga, Commodore, TRS80, Sinclair, Apple, among others) completely incompatible. BASIC[33] was the common language, although with different implementations. Then, in 1981, IBM created the PC – the first small computer to be able to run business applications - and in 1984, Apple released the Macintosh – the first popular graphical platform. The era of microcomputers started.

The natural evolution of the microcomputers, today's desktops are able to run powerful stand-alone applications, like word processing and multimedia production. They are normally based on Intel or Macintosh platforms and used by small businesses or home users. They are much more powerful when connected to one or more servers, via network or the Internet. They become clients, and can be used to prepare and generate requests that are processed by the servers, and then receive, format and display the results.

Most of the desktop market is dominated by computers originated from the IBM PC architecture, built from several suppliers (and even sold in individual parts) around Intel and AMD processors. Apple is also present – with a different architecture - traditionally in the publishing and multimedia productions.

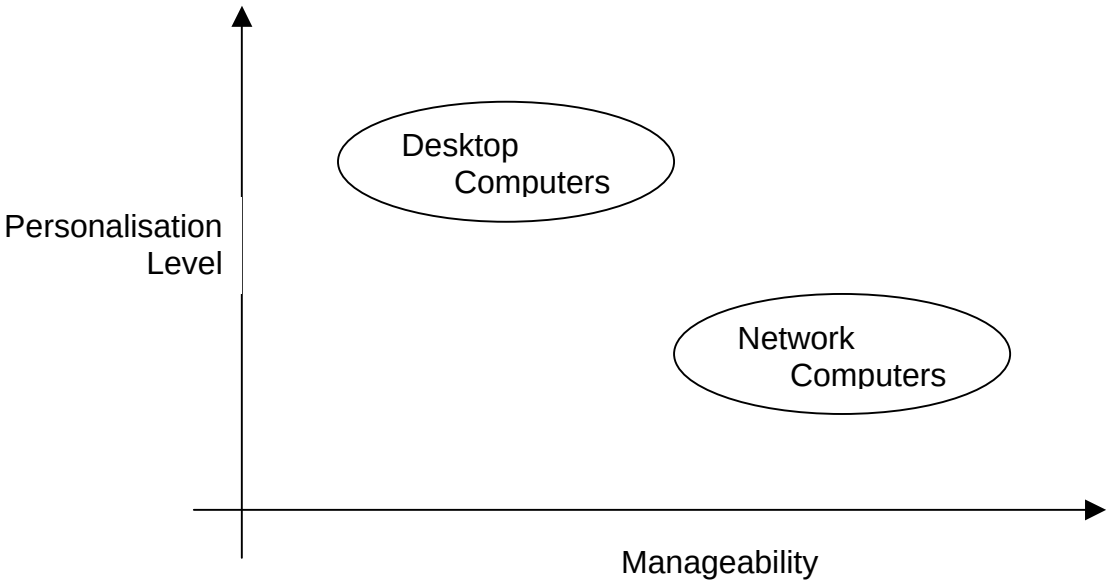


Figure 8 – Desktop x Network computers

[32] More information can be obtained on the internet: www-3.ibm.com/autonomic/
[33] BASIC stands for Beginner's All-purpose Symbolic Instruction Code, and is an easy-to-learn programming language, without a good logical structure.

- **Network Computing**

Recent evolutions from the desktops, the Network Computers (also called WebPC or NetPC) are also used as clients, with less power and more flexibility. Normally these computers have built-in software, which is able to connect to the network and fetch the needed applications from the servers, or simply use the applications running on the servers themselves.

The main objective is to allow shops and sales persons to work from anywhere in the world in the same way, with the same profile, even if changing the hardware interface. It also allows a better level of control over personalization, as all the information is always stored in the server, the client working simply as the interface.

Main suppliers are IBM and Compaq, with network computers working as PC clients, and Citrix with a proprietary client able to connect only to Citrix servers and obtain a perfect image of PC client applications via the network. Similar technology can be found under the names “thin clients” and “smart display”.

To support the development of network computing, the company Sun developed a new language (Java[34]), aiming to guarantee the independence of the hardware equipment or platform by the usage of Virtual Machines[35].

- **Pervasive Computing**

The idea is to connected small devices than laptops to the network. The PDAs are largely used today, and have the capacity to work in network, even if the communication costs avoid many to use them to send e-mails and to connect to the Internet.

With the decreasing size of the chips, and with the development of wireless protocols – like bluetooth and WiFi - some technology “hype-makers” are trying to convince the general public that they need every single appliance connected to the network. Although with some interesting applications, most of them are simply gadgets. With the current economic crisis, there is a low probability that such products will have a viable commercialisation soon.

Some companies are using these technologies in warehouse management systems, to reduce costs and increase the control on stock and product tracing. Also current is the usage of Linux in set-top boxes, able to select TV channels, save programs in hard disks for later viewing, filter channel viewings according to license keys.

See also chapter 2.3.4 *(Trend: Open Spectrum)*.

[34] See chapter 3.2.1 on page 61
[35] Source: bibliography 27

2.2. Operating Systems

As we already discussed, hardware is the equipment used to process information. An Operating System is the system software responsible to manage the hardware resources (memory, Input/Output operations, processors and peripherals), and to create an interface between these resources and the operator or end-user[36].

Originally, each hardware platform could run its own operating system family. This is not true any more. Here we are going to briefly analyse some examples of well-known operating systems, and present some historical facts related to standards, architectures and open source. Later we are going to discuss the initial dependence between the operating systems and the architecture, and how this has changed.

2.2.1. z/VM and z/OS

With a history started in 1964, IBM mainframe operating systems are currently z/VM and z/OS (a third one, VSE/ESA[37], is only kept for compatibility issues). They share the same architecture (currently called z/Architecture[38]) and their main characteristics are:

- **z/VM**[39] was originally built to ease the machine resources sharing by creating simulated computers (virtual machines), each running its own operating system[40].

- **z/OS**[41] - has always been the operating system used by large IBM systems, offering a great level of availability and manageability.

- **Architecture**

IBM has been the leader of the mainframe commercial market since the very beginning, with strong investments in R&D and marketing strategies, and with a determination to use common programming languages (like COBOL) in the conquest of programmers and customers. One key factor for IBM's success is standardization. In the 1950s, each computer system was uniquely designed to address specific applications and to fit within narrow price ranges. The lack of compatibility among these systems, with the consequent huge efforts when migrating from one computer

[36] For a more complete description of the operating systems concept and their history, please refer to http://www.wikipedia.org/wiki/History_of_operating_systems
[37] VSE/ESA stands for Virtual Storage Extended/ Enterprise System Architecture
[38] The "z" stands (according to IBM) for "zero down-time".
[39] VM stands for "Virtual Machine"
[40] The initial interest was to create a handful of virtual machines, to run DOS/VSE systems, and hundreds of machines given to the users for word processing and other applications with a low level of data sharing.
[41] Z/OS is still commonly called MVS – which stands for "Multiple Virtual Storage" - (despite of IBM marketing tentative of renaming it to OS/390 and now z/OS.

to another, motivated IBM to define a new ”architecture” as the common base for a whole family of computers. It was known as the System/360, announced on April 1964[42], which allowed the customers to start using a low-cost version of the family, and upgrade to larger systems if their needs grew. Scalable architecture completely reshaped the industry. This architecture introduced a number of the standards for the industry - such as 8-bit bytes and the EBCDIC character set[43] - and later evolved to the S/370 (1970), 370-XA (1981), ESA/370 (1988), S/390 (1990) and z/Architecture (2000). All these architectures were backward compatible, allowing the programs to run longer with less or no adaptation, while convincing the customers to migrate to the new machines and operating systems versions to profit from new technologies.

The standardization of the hardware platforms, by using a common architecture, helped to unify the efforts from the different IBM departments and laboratories spread over the world. The usage of a common set of basic rules was also fundamental in the communication among the hardware and software departments, which allowed the existence of the three different operating system families, able to run in the same hardware.

- **Relative Openness**

This architecture is proprietary (privately owned and controlled - the control over its use, distribution, or modification is retained by IBM), but its principles and rules are well defined and available to the customers, suppliers and even the competitors[44]. This openness allowed its expansion by:

- Hardware - Other companies (OEM - Original Equipment Manufacturer) started to build hardware equipment full compatible with the IBM architecture (thus allowing its use together with IBM hardware and software), and often with advantages on pricing, performance or additional functions. This gave a real boost on the IBM architecture, by giving the customer the advantage of having a choice, while enforcing the usage of its standards and helping it to sell even more software and hardware.

- Software – Initially IBM was dedicated to building operating systems, compilers and basic tools. Many companies (ISV - Independent Software Vendors) started to develop software to complement this basic package, allowing the construction of a complete environment. The end-user could then concentrate in the development of its applications, while buying the operating system and the complementary tools from IBM and the ISV.

- Services – As the IBM architecture became the standard for large systems, consulting companies could then specialize in this market segment, providing standard service offerings around the implementation of the hardware components, operating systems and software. With the recent increase on the demand for this type of service, mainly due to the Y2K problem, the Euro

[42] “The “360” in the name referred to all points of a compass to denote universal applicability, a wide range of performance and prices, and the “whole company” scope of the development effort” – See bibliography 14.
[43] EBCDIC is explained on page 80.
[44] See online version on http://publibz.boulder.ibm.com/epubs/pdf/dz9zr000.pdf

implementation and the outsourcing hype, most of the infrastructure services in large environments have been done by external consultants[45].

- **The negative aspects**

There's also a dark side: several times, IBM has been accused of changing the architecture without giving enough time to the competitors to adapt their hardware and software. Only when the competition from other high-end servers suppliers increased, IBM has been forced to review this policy, creating the concept of "ServerPac". The operating system started to be bundled with ISV software, giving the user a certain guarantee of compatibility.

A second problem is the complete ownership of the standards, by IBM. Even if other companies found good ways of implementing new technologies, thus improving the quality of the service or the performance, they always needed to adapt their findings to fit in the IBM architecture[46]. This limited the innovation coming from the OEM companies as they could rarely think about improvements that would imply architectural changes. On the other hand, the customers were never sure if the changes imposed by IBM were really needed, or simply another way of forcing them to upgrade the operating systems version or the hardware equipment with an obvious financial benefit for IBM.

A third aspect is related to the availability of the code source of the operating systems and basic software. Originally, IBM supplied the software with most part of the source code and a good technical documentation about the software's internal structures. This gave the customers a better knowledge about the architecture and the operating systems, allowing them to analyse problems independently, to drive their understanding beyond the documentation, and to elaborate routines close to the operating systems. After, they created the "OCO" (Object Code Only) concept, which persists today. The official reason behind this was the difficulty for IBM to analyse software problems, due to the increasing number of customers that started to modify the IBM software to adapt them to local needs, or to simply correct bugs. In fact, by hiding the source code IBM also avoided that customers discovered flaws and developed interesting performance upgrades. This also helped to decrease the knowledge level from the technical staff, which is limited today to follow the instructions given by IBM, when installing and maintaining the software. The direct consequence was a lack of interest for the real system programmers and students, who have been attracted to UNIX and Open Source environments. A positive consequence of the OCO policy is the quicker migration from release to release. Since the code is not modified, but instead APIs (Application Programming Interfaces[47]) can be used to adapt the system to each customer needs.

- **Impacts**

These factors were crucial to create a phenomenon called downsizing: Several customers, unhappy with the monopoly in the large systems, and with the high prices practiced by IBM and followed by the OEM and ISV suppliers, started to migrate their

[45] Even with the recent reduction in the consultancy expenses, most of the essential activities – related to the core business – have been "in-sourced back", but the infrastructure remains externalised.
[46] By the way, IBM was not interested in adapting its standards to improve the performance, as one of its major financial benefits was from selling larger machines.
[47] In the mainframe environment APIs are usually available in the form of "user exits".

centralized (also called enterprise) systems to distributed environments, by using midrange platforms like UNIX. In parallel (as it will be discussed on chapter 2.2.2), there were many developments of applications and basic software in the UNIX platforms, the most important around the Internet. To counterattack, IBM decided to enable UNIX applications to run in their mainframe platforms: MVS (the Open Edition environment, now called UNIX System Services) and VM (by running Linux as guest operating systems in virtual machines). IBM major effort today is to create easy bridges between those application environments, and to convince the customers to web-enable the old applications, instead of rewriting them.

Important is to notice that the standardisation on the hardware level was not abstracted to the operating system level. As mentioned above IBM always maintained three different operating systems, each one targeted to a different set of customers. This was not always what IBM desired, and in every implementation of a new architecture level (e.g. XA, ESA), the rumours were that the customers would be forced to migrate from VM and VSE to MVS, which would become the only supported environment. This is still true for the VSE (note that there's no z/VSE announced yet), but z/VM became a strategic environment allowing IBM to implement Linux in all hardware platforms.

Usually, the mainframe implementations are highly standardized. As the professionals in this domain understood the need for standards, the creation of company policies is done even before installing the systems.

2.2.2. UNIX®

"Unix was the distilled essence of operating systems, designed solely to be useful. Not to be marketable. Not to be compatible. Not to be an appendage to a particular kind of hardware. Moreover a computer running Unix was to be useful as a computer, not just a `platform' for canned `solutions'. It was to be programmable - cumulatively programmable. The actions of program builders were to be no different in kind from the actions of users; anything a user could do a program could do too...." (McIlroy - Unix on My Mind)

An analysed by Giovinazzo, "while it may seem by today's standards that a universal operating system like UNIX was inevitable, this was not always the case. Back in that era there were a great deal of cynicism concerning the possibility of a single operating system that would be supported by all platforms. (…) Today, UNIX support is table stakes for any Independent Software Vendor (ISV) that wants to develop an enterprise class solution"[48].

It may surprise some people but the conception of UNIX (MULTICS) started almost in the same time than the IBM System/360, and the first UNIX edition was released just after the IBM System/370, in November 1971[49]. MULTICS was developed by the General Electric Company, AT&T Bell Labs[50] and the Massachusetts Institute of Technology to allow many users to access a single computer simultaneously, allowing them to share data and the processing cost. Bell Labs abandoned the project and used some of MULTICS concepts to develop UNIX on a computer called DEC PDP-7, predecessor of the VAX computers[51].

- **Portability**

The "C" language has been created under UNIX and then - in a revolutionary exploit of recursion - the UNIX system itself has been rewritten in "C". This language was extremely efficient while relatively small and started to be ported to other platforms, allowing the same to happen with UNIX.

Besides portability, another important characteristic of the UNIX system was its simplicity (the C logical structure could be learnt quickly, and UNIX was structured as a flexible toolkit of simple programs). Teachers and students found on it a good way to study the very principles of operating systems, while learning UNIX more deeply, and quickly spreading UNIX principles and advantages to the market.

This helped UNIX to become the ARPANET (and later Internet) operating system by excellence. The universities started to migrate from proprietary systems to the new open environment, establishing a standard way to work and communicate.

[48] See bibliography 13, page 15.
[49] See bibliography 16, page 1-4.
[50] http://www.bell-labs.com/history/unix/
[51] See bibliography 5, page 23.

- **Distribution method**

Probably one of the UNIX most innovations was its original distribution method. AT&T could not market computer products so they distributed UNIX in source code, to educational institutions, at no charge. Each site that obtained UNIX could modify or add new functions, by creating a personalized copy of the system. They quickly started to share these new functions and the system become adaptable to a very wide range of computing tasks, including many completely unanticipated by the designers.

Initiated by Ken Thompson[52], students and professors from the University of California-Berkeley continued to enhance UNIX, creating the BSD (Berkeley Software Distribution) Version 4.2, and distributing it to many other universities. AT&T distributed their own version, and were the only able to distribute commercial copies.

In the early 80s the microchip and local-area network started to have an important impact on the UNIX evolution. Sun Microsystems used the Motorola 68000 chip to provide an inexpensive hardware for UNIX. Berkeley UNIX developed built-in support for the DARPA Internet protocols, which encouraged further growth of the Internet[53]. "X Window" provided the standard for graphic workstations. By 1984 AT&T started to commercialise UNIX. "What made UNIX popular for business applications was its timesharing, multitasking capability, permitting many people to use the mini- or mainframe; its portability across different vendor's machines; and its e-mail capability"[54].

- **The UNIX wars**

Several computer manufacturing companies[55] - like Sun, HP, DEC and Siemens - adapted AT&T and BSD UNIX distributions to their own machines, trying to seduce new users by developing new functions to benefit from hardware differences. Of course, the more differences between the UNIX distributions, more difficult to migrate between platforms. The customers started to be trapped, like in all other computer families.

As the versions of UNIX grew in number, the UNIX System Group (USG), which had been formed in the 1970s as a support organization for the internal Bell System use of UNIX, was reorganized as the UNIX Software Operation (USO) in 1989. The USO made several UNIX distributions of its own - to academia and to some commercial and government users –and then was merged with UNIX Systems Laboratories, to become an AT&T subsidiary.

AT&T entered into an alliance with Sun Microsystems to bring the best features from the many versions of UNIX into a single unified system. While many applauded this decision, one group of UNIX licensees expressed the fear that Sun would have a commercial advantage over the rest of the licensees.

[52] One of the creators of UNIX, working for the AT&T Bell labs. Biography in http://www.bell-labs.com/history/unix/thompsonbio.html, web site on http://www.cs.bell-labs.com/who/ken/.
[53] See chapter 2.3.3
[54] http://www.bell-labs.com/history/unix/business.html
[55] http://www.unix.org/images/chronology_big.gif

The concerned group, leaded by Berkeley, in 1988 formed a special interest group, the Open Systems Foundation (OSF), to lobby for an "open" UNIX within the UNIX community. Soon several large companies also joined the OSF.

In response, AT&T and a second group of licensees formed their own group, UNIX International. Several negotiations took place, and the commercial aspects seemed to be more important than the technical ones. The impact of losing the war was obvious: important adaptations should be done in the complementary routines developed, some functions – together with some advantages – should be suppressed, some hardware innovative techniques should be abandoned in the name of the standardization. When efforts failed to bring the two groups together, each one brought out its own version of an "open" UNIX. This dispute could be viewed two ways: positively, since the number of UNIX versions were now reduced to two; or negatively, since there now were two more versions of UNIX to add to the existing ones.

In the meantime, the X/Open Company – company formed in guise to define a comprehensive open systems environment - held the centre ground. X/Open chose the UNIX system as the platform for the basis of open systems and started the process of standardizing the APIs necessary for an open operating system specification. In addition, it looked at areas of the system beyond the operating system level where a standard approach would add value for supplier and customer alike, developing or adopting specifications for languages, database connectivity, networking and connections with the mainframe platforms. The results of this work were published in successive X/Open Portability Guides (XPG).

- **The Single UNIX Specification**[56]

In December 1993, one specification was delivered to X/Open for fast track processing. The publication of the Spec 1170[57] work as the proper industry supported specification occurred in October 1994. In 1995 X/Open introduced the UNIX 95 brand for computer systems guaranteed to meet the Single UNIX Specification. On 1998 the Open Group introduces the UNIX 98 family of brands, including Base, Workstation and Server. First UNIX 98 registered products shipped by Sun, IBM and NCR.

There is now a single, open, consensus specification, under the brand X/Open® UNIX. Both the specification and the trademark are now managed and held in trust for the industry by X/Open Company. There are many competing products, all implemented against the Single UNIX Specification, ensuring competition and vendor choice. There are different technology suppliers, which vendors can license and build their own product, all of them implementing the Single UNIX Specification.

Among others, the main definitions under this specification on version 3 - which is a result of IEEE POSIX, The Open Group and the industry efforts – are the definitions (XBD), the commands and utilities (XCU), the system interfaces and headers (XSH)

[56] From “The Authorized Guide to the Single UNIX Specification, Version 3” (http://www.unix-systems.org/version3/theguide.html) and “The Open Group – History and timeline (http://www.unix.org/what_is_unix/history_timeline.html).

[57] There were 1170 interfaces in the complete specification when the work was done (926 programming interfaces, 70 headers, 174 commands and utilities). There are now more than 1170 interfaces in the specification as the review process shaped the document accordingly.

and the networking services. They are part of the X/Open CAE (Common Applications Environment) document set.

In November 2002, the joint revision to POSIX® and the Single UNIX® specification have been approved as an International Standard[58].

- **Today**

The Single UNIX Specification brand program has now achieved critical mass: vendors whose products have met the demanding criteria now account for the majority of UNIX systems by value. UNIX-based systems are sold today by a number of companies[59].

UNIX is a perfect example of a constructive way of thinking (and mainly: acting!), and it proves that the academia and commercial companies can act together, by using open standards regulated by independent organisations, to construct an open platform and stimulate technological innovation.

Companies using UNIX systems, software and hardware compliant with the X/Open specifications can be less dependent of the suppliers. Migration to another compliant product is always possible. One can argue that, in cases of intensive usage of "non-compliant" extra features, the activities required for a migration can demand a huge effort.

[58] Designated as ISO/IEC 9945:2002, the joint revision forms the core of The Open Group's Single UNIX Specification Version 3 (IEEE 1003.1-2001, POSIX.1). More information on http://www.iso.ch/iso/en/commcentre/pressreleases/2002/Ref837.html

[59] The systems include Solaris® from Sun Microsystems, HP-UX® from Hewlett-Packard, AIX® from IBM, and Tru64 UNIX® from Compaq
Platform Vendors Supporting the Single UNIX Specification: Acer; Amdahl; Apple; AT&T GIS; Bull; Convex; Cray; Data General; Compaq; Encore; 88 Open; Fuji Xerox; Fujitsu Ossi; Hal; Hewlett-Packard; Hitachi; IBM; ICL; Matsushita; Mips ABI; Mitsubishi; Motorola; NEC; Novell/USL; Oki; Olivetti; OSF; PowerOpen; Precision RISC; Pyramid; SCO; Sequent; Sequoia; Sharp; Siemens-Nixdorf; Silicon Graphics; Sony; Sparc International; Stratus; Sun Microsystems; Tadpole; Tandem; Thompson/Cetia; Toshiba; Unisys; Wang Labs.
ISVs and User Organizations Supporting the Common API Specification: AutoDesk; Banyan; Bellcore; Bentley; Cadence; Cadre; Chorus; Computer Associates; DHL; EDS Unigraphics; Frame Tech; Informix; Island Software; Lachman Tech; Locus; Lotus; McDonald's; Mentor; Oracle; Pencom Systems; SDRC; Software AG; Shell Oil; Veritas; Wal-Mart; WordPerfect.
Source: http://www.unix.org/what_is_unix/single_unix_specification.html

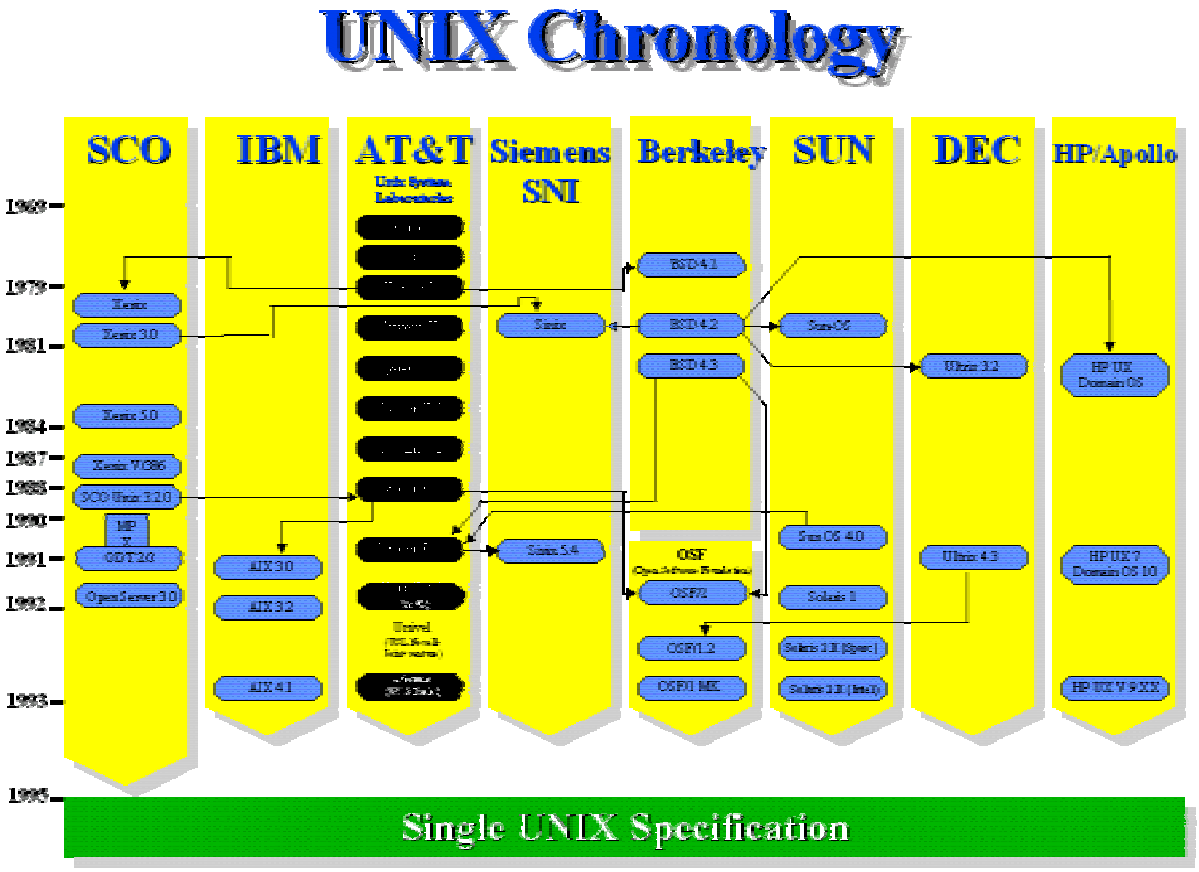


Figure 9 – UNIX Chronology[60]

[60] Source: The Open Group (http://www.unix.org/images/chronology_big.gif)

2.2.3. Linux

"Every good work of software starts by scratching a developer's personal itch" – Eric Raymond

This discussion is not completely separated from the previous one. Developed by Linus Torvalds, Linux is a product that mimics the form and function of a UNIX system[61], but is not derived from licensed source code. Rather, it was developed independently by a group of developers in an informal alliance on the net (peer to peer). A major benefit is that the source code is freely available (under the GNU copyleft), enabling the technically astute to alter and amend the system; it also means that there are many, freely available, utilities and specialist drivers available on the net.

- **Brief Hacker History**

The hackers – not to be confounded with the "crackers"[62] – appeared in the early sixties and define themselves as people who "program enthusiastically"[63] with "an ethical duty (…) to share their expertise by writing free software and facilitating access to information and computing resources wherever possible"[64]. After the collapse of the first software-sharing community[65] - the MIT Artificial Intelligence Lab - the hacker Richard Stallman quit his job at MIT in 1984 and started to work on the GNU[66] system. It was aimed to be a free operating system, compatible with UNIX. "Even if GNU had no technical advantage over UNIX, it would have a social advantage, allowing users to cooperate, and an ethical advantage, respecting the user's freedom"[67]. He started by gathering pieces of free software, adapting them, and developing the missing parts, like a compiler for the C language and a powerful editor (EMACS). In 1985, the Free Software Foundation[68] has been created and by 1990, the GNU system was almost complete. The only major missing component was the kernel.

In 1991, Linus Torvalds developed a free UNIX kernel using the FSF toolkit. Around 1992, the combination of Linux and GNU resulted in a complete free operating

[61] Recent versions of Glibc include much functionality from the Single UNIX Specification, Version 2 (for UNIX 98).
[62] In the mid-eighties the term "hacker" started to be misused to address the computer criminals. The real hackers created the term "cracker" to apply to the virus writers and intruders, but the confusion still exists in the media and general public.
[63] The Jargon File, s.v. Hacker (www.tuxedo.org/~esr/jargon)
[64] The Jargon File, s.v. Hacker Ethic
[65] See bibliography 5, page 53.
[66] The name GNU was chosen following a hacker's tradition, as a recursive acronym for "GNU's Not UNIX"
[67] Richard Stallman - bibliography 5, page 61.
[68] FSF – A tax-exempt charity for free software development, distribution and services.

system, and by late 1993, GNU/Linux could compete on stability and reliability with many commercial UNIX versions, and hosted more software.

Linux was based on good design principles and a good development model. By opposition to the typical organisation – where any complex software was developed in a carefully coordinated way by a relatively small group of people – Linux was developed by a huge numbers of volunteers coordinating through the Internet.

Portability was not the original goal. Conceived originally to run on a personal computer (386 processor), later on some people ported the Linux kernel to the Motorola 68000 series – used in early personal computers – using an Amiga computer. Nevertheless, the serious effort was to port Linux to the DEC Alpha machine. The entire code has been reworked in such a modular way that future ports were simplified, and started to appear quickly.

This modularity was essential for the new open-source development model, by allowing several people to work in parallel without risk of interference. It was also easier for a limited group of people – still coordinated by Torvalds – to receive all modified modules and integrate into a new kernel version[69].

- **To be or not to BSD**

Besides Linux, there are many freely available UNIX and UNIX-compatible implementations, such as OpenBSD, FreeBSD and NetBSD. According to Gartner, "Most BSD systems have liberal open source licenses, 10 years' more history than Linux, and great reliability and efficiency. But while Linux basks in the spotlight, BSD is invisible from corporate IT"[70]. BSDI is an independent company that markets products derived from the Berkeley Systems Distribution (BSD), developed at the University of California at Berkeley in the 60's and 70's. It is the operating system of choice for many Internet service providers. It is, as with Linux, not a registered UNIX system, though in this case there is a common code heritage if one looks far enough back in history. The creators of FreeBSD started with the source code from the Berkeley UNIX. Its kernel is directly descended from that source code.

- **(Dis)united Linux**

Recursion or fate, the fact is that Linux history is following the same steps than UNIX. As it was not developed using the source from one of the UNIX distributions, Linux have several technical specifications that are not (and will probably never be[71]) compliant with UNIX98. Soon, Linux started to be distributed by several different companies worldwide – SuSE, Red Hat, MandrakeSoft, Caldera International / SCO and Conectiva, to name some –, bundled together with a plethora of open source software, normally in two different packages - one aiming at home users and another

[69] A long-time kernel maintainer was Alan Cox, who transferred this task to the Brazilian Marcelo Tosatti (source: ZDNet July, 11 2002)
[70] See bibliography 40.
[71] The theoretical paths needed to be followed by Linux to get a certification are exploited by Ian Nandhra on http://lwn.net/1998/0611/standardseditorial.html

for companies. In 2000, two standardization groups appeared - LSB[72] and LI18NUX[73] - and have incorporated under the name Free Standards Group, “organized to accelerate the use and acceptance of open source technologies through the application, development and promotion of interoperability standards for open source development”[74]. Caldera, Mandrake, Red hat and SuSE currently have versions compliant with the LSB certification[75].

Still in an early format, the current standards are not enough to guarantee the compatibility among the different distributions. In a commercial maneuver – to reduce the development costs – SCO, Conectiva, SuSE and Turbolinux formed a consortium called UnitedLinux, a joint server operating system for enterprise deployment. The software and hardware vendors can concentrate on one major business distribution, instead of certifying to multiple distributions. This is expected to increase the availability of new technologies to Linux customers, and reduce the time needed for the development of drivers and interfaces. Consulting companies can also concentrate the efforts to provide more and new services.

However, Red hat – the dominant seller in the enterprise market segment – has not been invited to join the UnitedLinux family before its announcement. Few expect it to join now. Also missing are MandrakeSoft and Sun Microsystems[76].

With the help from important IT companies, it’s expected from the Free Standards Group the same unifying role than the one played by the Open Group, which was essential for the UNIX common specification[77]. This is very important to give more credibility for the companies willing to seriously use Linux on their production environments. As in the UNIX world, the Linux sellers could still keep their own set of complementary products, which would be one important factor to stimulate the competition. The distributions targeting the home users can continue separated, by promoting the diversity needed to stimulate creativity, and by letting the natural selection chose the best software to be elected for enterprise usage. Darwin would be happy. Also would the users.

- **Servers and desktops[78]**

Linux already proved to be a reliable, secure and efficient operating system, ranging from low entry up to high-end servers. It has the same difficulty level than other UNIX platforms, with the big advantage of being economically affordable – almost

[72] Linux Standard Base (http://www.linuxbase.org/)
[73] LI18NUX (Linux Internationalization Initiative) has later changed its name to OpenI18N (Open Internationalization Initiative). This was done to better reflect the groups open source activities that go beyond Linux (http://www.openi18n.org/charter/). The peculiar designation derives from the widely used abbreviation I18N: the letters "i" and "n" of "internationalization" are separated by 18 letters.
[74] Source: http://www.openi18n.org/press/FreeStandards/
[75] Source: http://www.opengroup.org/lsb/cert/cert_prodlist.tpl (status: November 2002).
[76] Source: ZDNet June 3, 2002 (http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2868859,00.html)
[77] See item “The Single UNIX Specification” on page 27.
[78] I have based my analysis in this paragraph in articles from the DataNews publication - about Linux experiences in Belgian and European companies - in several discussions with people using Linux as desktops and servers, and in my own experience by installing Linux at home. An interesting article has been written by Don Soegaard and can be found in bibliography 39.

everybody can have it at home, and learn it by practice. However, is it technically possible for everyone to install it and use it?

The main argument supporting all the open source software is that a good support can be obtained from service providers, besides the help from the open source community, by using Internet tools like newsgroups and discussion lists. Although, our question may have two different answers: One for servers, other for clients.

It is certainly possible for companies to replace existing operating systems with Linux, and the effort needed is mainly related to the conversion of the applications than to the migration of the servers or the operating systems. Several companies have been created using Linux servers to reduce their initial costs, and they worked with specialized Linux people from the beginning. Most of the companies replacing other platforms by Linux already have a support team, with a technological background, responsible by the installation and maintenance of the servers. Part of this team has already played with Linux at home and has learnt the basic knowledge. The other part is often willing to learn it, compare with the other platforms, and to discover what it has that is seducing the world.

The desktop users have a completely different scenario. This is because Linux is based on a real operating system, conceived to be installed and adapted by people with a good technical background, and willing to dedicate some time for this task. The main goal of most of the people (except probably the hackers themselves) installing Linux at home is to learn it, to play around, to compare with the other systems. The Linux main objective (by using the recursion so dear to the hackers) appears to be Linux itself. In its most recent releases, even if the installation process is not difficult as before, simple tasks as adding new hardware, playing a DVD, writing a CD or installing another piece of software, can quickly become nightmares. Linux can't yet match Windows on plug-and-play digital media or the more peripheral duties.[79] One should continue to be optimist and hope that the next versions will finally be simpler. The open source community is working on it, via the creation of the Desktop Linux Consortium[80].

- **Ubiquity … and beyond!**

Linux is considered the only operating system that will certainly run on architectures that have not yet been invented[81]. Four major segments of Linux key marketplaces are:

- Workload consolidation - One of the capabilities of Linux is that it makes it very easy to take distributed work and consolidate it in larger servers. Mainframes and some large servers support hundreds of thousands of virtual machines, each one running one copy of Linux.

- Clusters – Linux performs very well when connected in clusters (due to its horizontal scalability), like the big supercomputing clusters existing in universities and research labs.

[79] Source : bibliography 35
[80] Source: Datanews n°6, 14/02/2003, page 3
[81] This paragraph has been adapted fro, a study performed by IBM, available at (http://www.ibm.com/news/us/2001/08/15.html)

- Distributed enterprise – The central management of geographically distributed servers is among the chief growth areas for Linux.

- Appliances –Linux is found as an embedded operating system in all kinds of new applications, like major network servers, file and print servers, and quite a number of them new kinds of information appliances. It's very cost effective, reliable and fast. In addition, the open standards facilitate their operations.

2.2.4. Windows

The first Microsoft's product – in 1975 – was BASIC[82], installed in microcomputers used by hobbyists. In 1980, IBM started to develop the personal computer (PC) and invited Microsoft to participate in the project by creating the operating system. Microsoft bought, from a small company called Seattle Computer, a system called Q-DOS[83], and used it as a basis for MS-DOS, which became the operating system of choice, distributed by IBM as PC-DOS.[84] By 1983, Microsoft announced their planning to bring graphical computing to the IBM PC with a product called Windows. As Bill Gates explains, "at that time two of the personal computers on the market had graphical capabilities: The XEROX Star and the Apple Lisa. Both were expensive, limited in capability, and built on proprietary hardware architectures. Other hardware companies couldn't license the operating systems to build compatible systems, and neither computer attracted many software companies to develop applications. Microsoft wanted to created an open standard and bring graphical capabilities to any computer that was running MS-DOS"[85].

- **First steps**

Microsoft worked together with Apple during the development of the Macintosh, and created for this platform their two first graphical products: a word processor (Microsoft Word) and a spreadsheet (Microsoft Excel). Microsoft also cooperated with IBM during the development of OS/2, which was aimed to become the graphic replacement for MS-DOS.

Using this knowledge, and a lot of ideas from the XEROX research, Microsoft release the first version of Windows in November 1985[86]. The impact of OS/2 was very limited, as it did not provide full compatibility with the old MS-DOS software. On the other hand, Windows was a large success mainly because it supported the existing MS-DOS character-oriented applications, in parallel with exploiting the potentials from the new graphical software.

Application support for Windows was initially sparse. To encourage earlier application support, Microsoft licensed a free of charge runtime version of Windows to developers, which made their programs available to end-users. This runtime version allowed use of the application although it did not give the benefits that the full Windows environment provided. With version 3.1, Windows acquired maturity, reducing the incredible number of bugs - mainly related with the user interface, and

[82] Microsoft didn't create BASIC. It developed the interpreter able to run in microprocessors Intel 8080 (See bibliography 17). BASIC was created in the Dartmouth College by John Kemeny and Thomas Kurtz, in 1964 (See bibliography 18, page 29)
[83] Q-DOS stands for "Quick and Dirty Operating System". Source: Bibliography 18.
[84] The first IBM PC actually shipped with a choice of three operating systems: PC-DOS, CP/M-86 and the UCSD Pascal P-system. Using strong licensing strategies, Microsoft convinced IBM to abandon the other two. See bibliography 17, page 49.
[85] In bibliography 17.
[86] Complete list of versions and dates in http://www.computerhope.com/history/windows.htm

with the constant need of rebooting the system - and with the wide availability of graphic applications. Its popularity blossomed with the release of a completely new Windows software development kit (SDK) - which helped software developers focus more on writing applications and less on writing device drivers – and with the widespread acceptance among third-party hardware and software developers. The bugs are persistent, and today are the main source of virus by opening security flaws.

- **Different families with different editions**

Microsoft decided to keep the compatibility with the MS-DOS applications to avoid losing customers. However, this also avoided Windows desktop systems to be stable, and reduced the pace of innovation. Microsoft then started to develop a completely new desktop system from scratch, oriented to business customers that needed more stability. After 1993 two different families of desktop operating systems co-existed: the Windows 3x (that later evolved to Windows 9x and Windows Millennium), descended from MS-DOS and the Windows NT (that later evolved to Windows2000), completely built around the graphic interfaces and new hardware features. In 2001, Microsoft abandoned the 3x family and used the base code from NT to create the Windows XP, with two different packages and prices: one “edition” aimed for home users, another for professional use[87].

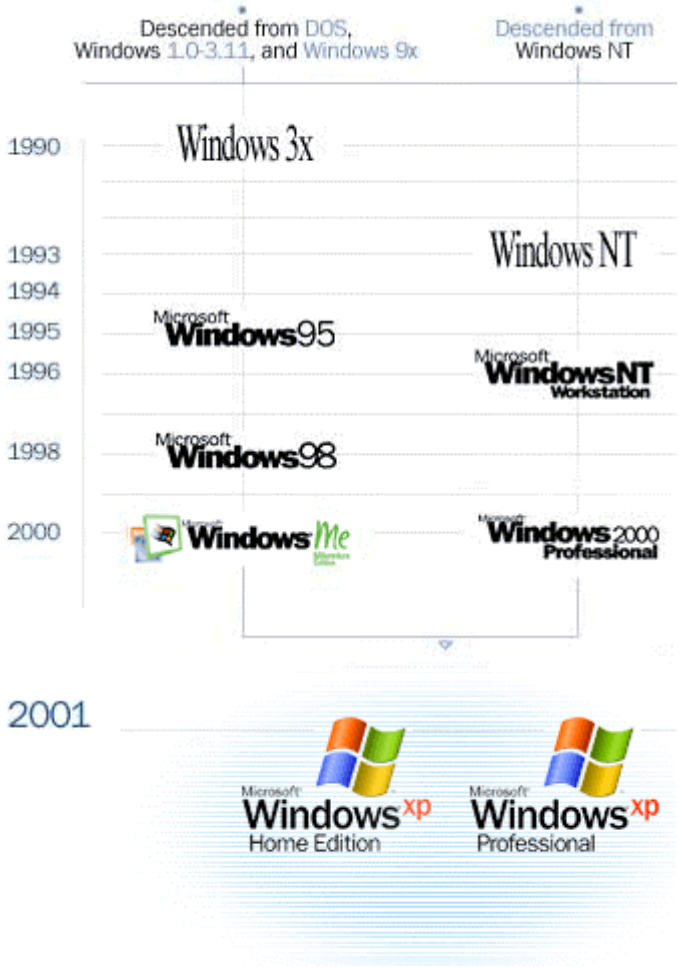


Figure 10 – Windows client evolution[88]

[87] See the timeline on http://www.microsoft.com/windows/winhistoryprographic.mspx
[88] Source: www.microsoft.com

In July 1993, Microsoft inaugurated its Windows server family with the release of Windows NT Advanced server 3.1[89]. It was designed to act as a dedicated server (for tasks as Application, Mail, Database and Communications Server) in a client/server environment, for Novell NetWare, Banyan VINES, and Microsoft networks. The next releases improved the connectivity with UNIX environments (3.5), integrated a web server (4.0) and consolidated the web facilities with ASP (2000). To follow is the Windows .NET server[90], which aims to better exploit the XML Web services and the .NET framework. The different editions for the server families are Standard, Enterprise, Datacenter and Web[91].

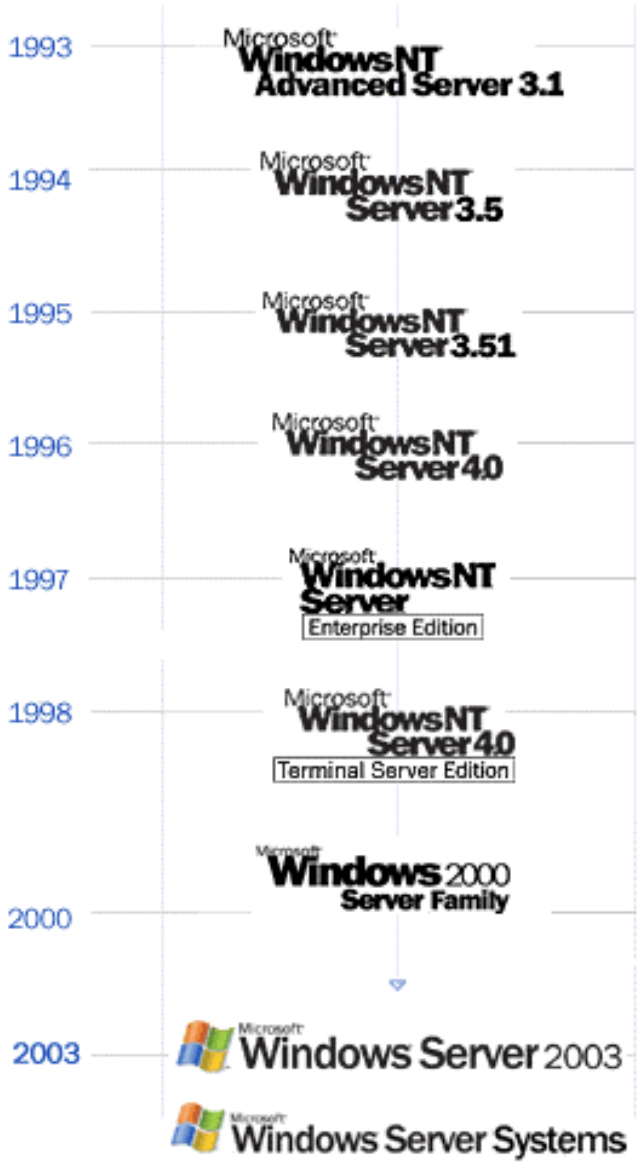


Figure 11 – Windows server evolution[92]

[89] Although the first server operating system from Microsoft was LAN Manager.
[90] Announced for final release in late 2002.
[91] See http://www.microsoft.com/windows/winhistoryserver.mspx
[92] Source: www.microsoft.com

- **Easy-to-use, easy-to-break**

The main advantage of the windows families – over Linux and UNIX - is their easy installation and operation. As we saw, Microsoft copied the concepts from Apple Macintosh and always based their windows products in graphic interfaces, with intuitive icons opening the doors for all the tasks, from running the applications to configuring the hardware and changing the software options, colours and sounds. This seems to be a major benefit for end users, that don't need to bother with learning the concepts of operating systems to be able to write a letter, scan some photos, mix everything and print. The problems start to appear when something changes, like the version of the operating system, a new hardware component or simply when an inoffensive application is installed or removed from the computer. These tasks often need an intervention from a specialist, or a complete reinstallation of the windows system. This happens even with the recent XP releases, despite Microsoft publicity stating the opposite.

Finally, the main technical advantages disappear, and only the market dominance, compatibility of the document formats, and marketing strategies convince the user to keep the windows system, rather than installing Linux or buying a Macintosh system.

- **Embrace and Extend**

Microsoft strategy always consisted in identifying the good opportunities, react quickly and offer a solution (often without taking care of the quality) to conquer the market and later improve it to keep the customers. Again, as it was the case for IBM with the mainframes, marketing strategies revealed to be more important than technological features, paving the way for "de facto" standards.

Windows is extremely important for Microsoft's marketing strategies, serving as a Trojan horse to introduce "free" Microsoft programs. This happened in 1995, when Internet Explorer quickly gained an impressive market place in a short time – again, instead of developing a concept and a product[93], Microsoft simply acquired Mosaic from the company Spyglass and make quick modifications transforming it into Internet Explorer. After, with the development of extensions to the standard HTML language, it almost eliminated Netscape from the market.

This drive us to the conclusion that Microsoft practices aim to conquer a complete monopoly – not only for operating systems, but also for applications, appliances, the internet and content[94] – through the complete elimination of the competition. This has already been analysed by the American and European justice departments.

- **Hardware Independence**

One of the main reasons behind Windows' success is the independence from the hardware supplier. If the processor chip is usually built by Intel and AMD, the computers can be manufactured by any company following the specifications from the PC architecture, or even assembled at home, by using standard components

[93] Microsoft only created a "Research" department in 1995 (source: bibliography 18).

[94] Bill Gates dedicated the chapters 4 to 7 to explain his vision about the future of Applications, Appliances, The Internet, and the Content, and how he expects Microsoft to dominate the realms of technology and social life by the use of "Killer Applications" (See bibliography 17).

available in any electronic shop. On the other hand, the Apple Macintosh and IBM PS/2 architectures force the customer to buy the hardware and software from the same supplier. There's no compatibility with other vendors, thus the cost remains higher than the PC/Windows platforms, which benefit from the competition and the higher market share to decrease the production costs.

Yet, the windows systems remain dependent of the PC architectures, and this may be a serious disadvantage by opposition to the portability offered by Linux systems. This is important to understand what is behind the strategies adopted by the Microsoft competitors, to be studied on the next chapter.

- **Social Organization**

One important difference between the proprietary MS-DOS and Windows families and the open systems is not technical, but social. There is not such a concept as Windows hackers[95]. Windows environments were conceived to run alone, not in networks. The UNIX systems have been built with strong networking facilities, since their first versions, what allowed creating a large community that used, analysed, developed and cooperated for its improvements. Moreover, the UNIX systems appeared in the academic world, which favoured this community to be extremely creative and looking for technological innovations. This concept, also known as peer-to-peer (P2P), obtained its climax with Linux, which has entirely constructed (and still evolves) through the Internet.

As well defined by Eric Raymond[96], "the MS-DOS world remained blissful ignorant of all this. Though those early microcomputer enthusiasts quickly expanded to a population of DOS and Mac hackers order of magnitude greater than that of the 'network nation' culture, they never become self-aware culture themselves. The pace of change was so fast that fifty different technical cultures grew and died as rapidly as mayflies, never achieving quite the stability necessary to develop a common tradition of jargon, folklore and mythic history. (…) The fact that non-UNIX operating systems don't come bundled with development tools meant that very little source was passed over them. Thus, no tradition of collaborative hacking developed.".

Complementary, as Himanen explains, "Bill Gates (…) gained hacker respect by programming his first interpreter of the BASIC programming language (…) [but] in Microsoft's subsequent history, the profit motive has taken precedent over the passion"[97].

[95] Except for the creators of virus and cyber-intruders.
[96] See bibliography 5, page 26
[97] See bibliography 4, page 56.

2.2.5. Other Operating Systems

- **Apple Mac/OS**

Released in 1984 and integrating the hardware and software into the same platform, the Apple Macintosh was the first to have good graphical interfaces to perform all the basic functions. Apple refused (until 1995) to let anyone else make computer hardware that would run it, limiting its success and avoiding it to become the standard for graphical platforms.

The technical comparison with PC and Windows is not easy to be done, and the eternal discussion is far from coming to an end. Apple addicts always considered the Mac as a more stable platform, and with best results with multimedia applications. The counterargument is that the modern windows / Intel systems improved their graphic and sound systems, with the help from external cards, and could produce the same quality. The reality today is that the Mac systems are more expensive, and are still preferred by designers and artists to produce graphic, music and images.

The new version, MacOsX, is based in the UNIX operating system OpenStep, and implement Vector-based graphic interfaces.[98]

- **IBM OS/2**

It's surprising that IBM is still maintaining versions of their OS/2 Warp operating system. In 1984, after having realized the real market potentials of personal computing, IBM decided to create a close platform (PS/2) with a stronger architecture than the IBM/PC – with mainframe customers in mind - and a solid operating system (OS/2), to be able to run professional applications and compete with the DOS, Windows and Macintosh families. The smaller prices of the PC clones, which already had a large installed base, and were sold with an incorporated DOS/Windows system, avoid this to happen. Contrarily to IBM hopes, the price was a more important factor than reliability, and the OS/2 platform has never been largely used.

- **IBM OS/400**

The only one that can unequivocally be called a "minicomputer" operating system, the OS/400 is the evolution of the IBM midrange System/36 and System/38 systems, implemented on PowerPC chips. Despite the lack of good technical reasons to keep both Unix and OS/400 platforms, the large installed base of these systems is able to guarantee their continuity.
Additionally, the current OS/400 gives the users the ability to run Java, Windows and UNIX applications (via PASE - Portable Application Solutions Environment which supports a subset of the AIX environment).

[98] See bibliography 18, page 160.

2.2.6. Classification

Considering two major criteria discussed in the previous paragraphs – scalability and openness – the operating systems can be classified by using the following chart:

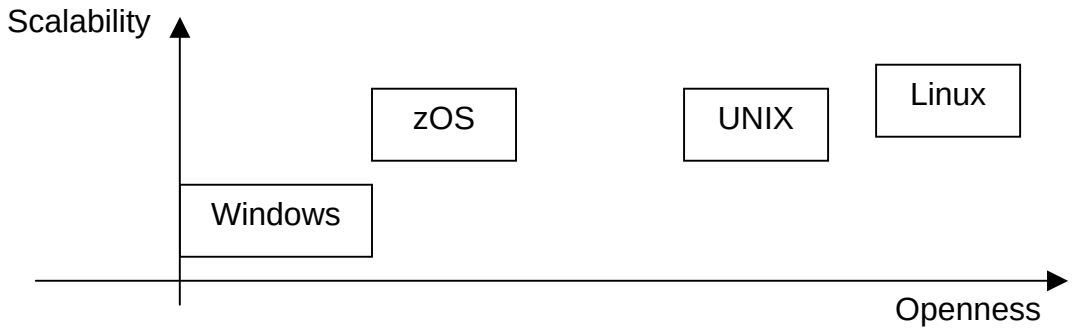


Figure 12 – Operating systems – Openness and scalability

Scalability is the ability of increasing the system throughput by adding processors or new systems, with a low impact in the efforts for managing the system (license costs and human resources). Linux outperform the other systems due to its capacity of running in virtually all hardware platform and machine sizes, its reduced cost, the openness of its source code and the usage of open standards and protocols.

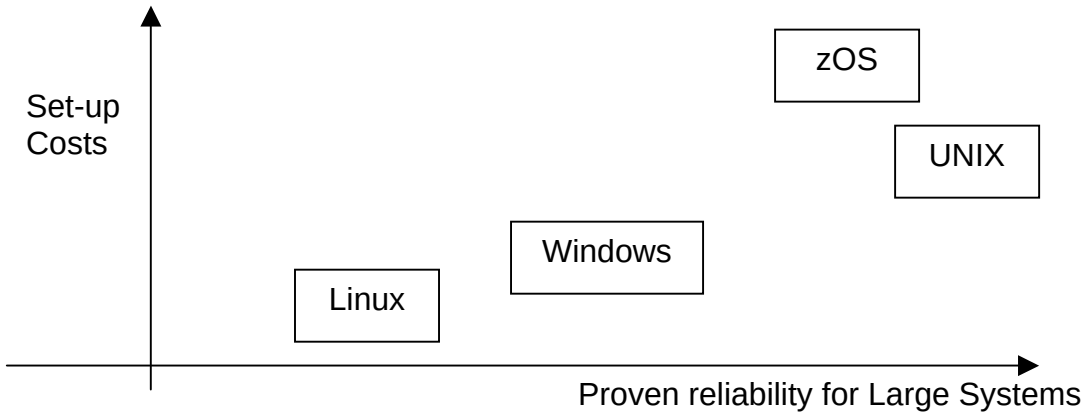


Figure 13 – Operating systems – set-up costs and proven reliability

The number of installed mainframe and UNIX systems for critical systems in large companies, in addition to the decades of experience and fine tuning, help these systems to have more confidence from IT specialists. The reduced initial setup costs[99] for windows and Linux platforms justify their choice when price is an important element. Goldman Sachs considers Linux as “Enterprise Class”, and estimates that Linux will soon replace windows in medium-large servers[100]. The cost analysis and comparisons will be detailed on chapter 6.4.

[99] The TCO (Total Cost of Ownership) is more difficult to be measured, and contradictory analysis can be found today, which difficult an independent comparison. See a detailed analysis of costs later in this document, on pages 103-109.
[100] Source: DataNews n°04, 31/1/2003, Page 26

2.3. Communication

The co-existence of different standards on the hardware and software domains may stimulate the competition, the innovation and the market. For the network infrastructures, the absence of widely accepted standard and protocols can simply avoid the communication to happen in a global way.

One of the first problems identified when the Internet ideas started to flourish was “to understand, design, and implement the protocols and procedures within the operating systems of each connected computer, in order to allow the use of the new network by the computers in sharing resources.”[101]

The free exchange of information via the Internet, independently of the hardware or operating system we may use, is possible due to the previous definition of standards, which we may classify in three different tiers: The communication via the network, the addressing of each node and the content formatting and browsing. Standard bodies are responsible for their definition, publication and conformity. Let us discuss the communication and addressing tiers. The content will be discussed in details on the chapter 4 (page 80).

2.3.1. Network

As we saw in chapter 2, the first mainframes and minicomputers performed very specific tasks, independent of any other computer, and the users and operators were connected locally, normally in the same building than the machines. When distant systems started to connect with each other, and to remote users, clear conventions needed to be established.

- **The OSI model**

The OSI reference model was developed by the International Organization for Standardization (ISO) in 1984, and is now considered the primary architectural model for inter-computer communications.

OSI is a conceptual model composed of seven layers, each specifying particular network functions. It only provides a conceptual framework. Several communication protocols have been developed using it as a reference, allowing their interoperability.

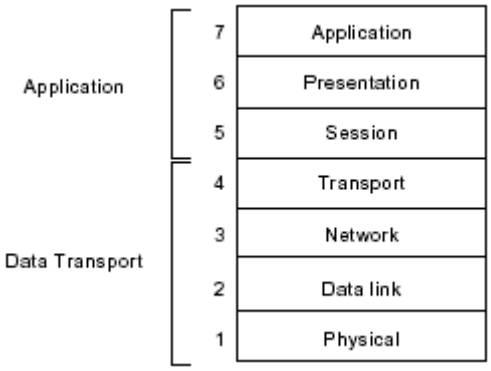


Figure 14 – The OSI model - Seven layers and two categories

[101] ARPA not draft, II-8

The seven layers of the OSI reference model can be divided into two categories: the upper layers (which deal with application issues and generally are implemented only in software) and the lower layers (which handle data transport issues).

- **The Internet Protocols**

In the mid-1970s, the Defense Advanced Research Projects Agency (DARPA) started to develop a packet-switched network that would facilitate communication between dissimilar computer systems at research institutions. The result of this development effort was the Internet protocol suite[102] that contained several basic network protocols. Among others, Transmission Control Protocol (TCP) and the Internet Protocol (IP) are the two best known and commonly used together[103].

One of the main reasons of their quick adoption by the research institutions (and later the whole market) was their openness. The Internet Protocol suite is nonproprietary, born in an academic environment close to the UNIX developments, and soon both work in synergy. TCP/IP was included with Berkeley Software Distribution (BSD) UNIX to become the foundation of the Internet[104].

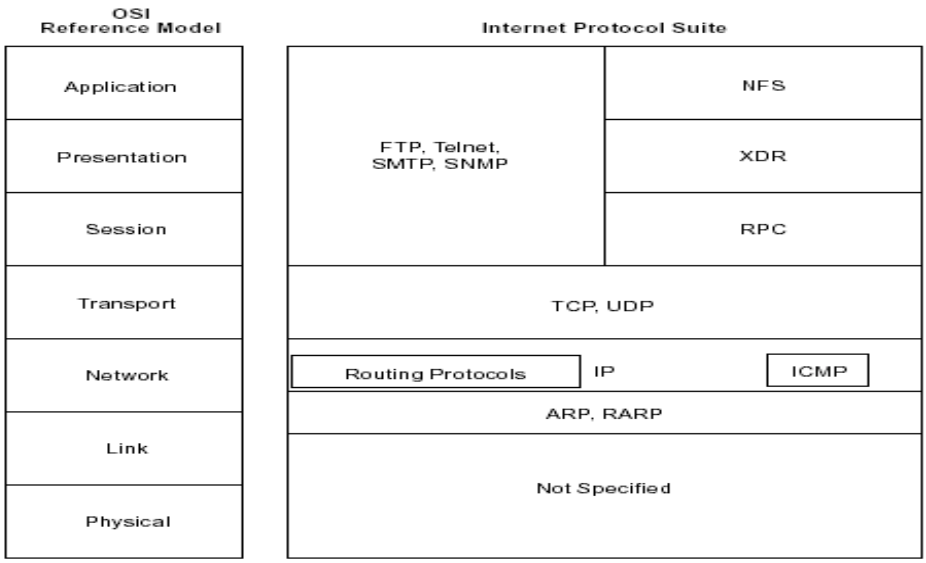


Figure 15 – Correspondence between the OSI layers and some Internet Protocols.

- **SNA**

In 1974, IBM introduced a set of communications standards, called SNA[105], which was extremely convenient to the mainframes hierarchical topologies. Like other IBM definitions for hardware and operating systems architectures, it is a proprietary protocol, but clearly defined and open for the use of other manufacturers. This allowed IBM and many other companies to provide servers communication hardware and programming products, which could be connected by the usage of common communication definitions, hardware specifications and programming conventions.

[102] Which was completed in the late 1970s
[103] Source: bibliography 44
[104] The TCP/IP protocols and simple applications were integrated into UNIX 4.2BSD, release in August 1983. Source: Bibliography 5, page 38.
[105] SNA stands for Systems Network Architecture

The administration of a SNA network is centralized, what improves its security while reducing its flexibility.

IBM tried to evolve SNA to an open-standard known as High-Performance Routing (HPR), without success. SNA is still used in the core mainframe connections, although it is being replaced by TCP/IP in distant connections, due to pressures of intranets and the Internet, and for more flexibility.

2.3.2. Addressing

A set of techniques is used to allow the identification of each single machine in the world, to allow the Internet packets to be distributed efficiently. Each computer with capability to be connected to the Internet receives from its manufacturer a single and unique address (called MAC address[106]). When a computer connects to an Internet or intranet server, it receives a network address[107].

As the IP addresses are quite difficult to remember, a domain name is created for important servers (like web or FTP servers)[108]. A hierarchy of systems have been created to locate a domain name. It is called DNS[109]. The intelligence behind the DNS servers is provided by the open source program BIND, and soon the "Domain Wars" started to decide who would control the domains.

The conflicts have been solved by a policy specifying the codification of roles in the operation of a domain name space:

- Registrant - The entity that makes use of the domain name.
- Registrar - The agent that submits change requests to the registry on behalf of the registrant.
- Registry - The organization that has edit control of the name space's database.

One non-profit organization has been created, ICANN[110], which the role is to oversee administer Internet resources including Addresses (by delegating blocks of addresses to the regional registries), Protocol identifiers and parameters (by allocating port numbers, etc.) and Names.

[106] The Media Access Control address has 48 bits and it is composed by the identification of the manufacturer (OUI - Organizationally Unique Identifier) and a serial number.
[107] Also known as TCP/IP address, and it may be fixed or dynamically assigned. It is something like 62.205.73.168. Web servers have fixed allocated network addresses. Google, for example, have the address 216.239.55.101 for its Belgian server.
[108] Following our previous example, the domain for Google Belgium is www.google.be.
[109] The "Domain Name System" has been created in 1983 by Paul Mockapetris (RFCs 1034 and 1035) and modified, updated, and enhanced by a myriad of subsequent RFCs. Source: bibliography 48
[110] ICANN stands for Internet Corporation for Assigned Names and Numbers. See chapter 11 for more info.

2.3.3. The Internet

- **History**

The Internet is the result of many technologies developing in parallel, around the Advanced Research Projects Agency (ARPA, today called DARPA[111]) of the U.S. Department of Defense. Motivated by the launching of the first sputnik, ARPA started to design a communications system invulnerable to nuclear attack. It was the ARPANET[112] project, based on packet switching communication technology. “The system made the network independent of command and control centers, so that message units would find their own routes along the network, being reassembled in coherent meaning at any point in the network”[113].

Important is to notice that, since the beginning, the concept around the network had stronger cultural and social than technical motivations: "The ARPA theme is that the promise offered by the computer as a communication medium between people, dwarfs into relative insignificance the historical beginnings of the computer as an arithmetic engine."[114]

J.C.R. Licklider[115] perceived the spirit of community created among the users of the first time-sharing systems. He envisioned an “'Intergalactic Network”, formed by academic computer centers. His ideas took almost one decade to become reality: On October 25, 1969 a host-to-host connection has been successfully established between the University of California Los Angeles (UCLA) and the Stanford Research Institute (SRI), creating the network roots for the Internet.

In the same time – as we already discussed – UNIX was born and the Internet Protocols have been defined. In 1973, 25 computers were connected using the ARPANET structure[116], opened to the research centers cooperating with the US defense Department. Other networks - using the ARPANET as the backbone communication system – started to appear, aimed to connect scientists from all disciplines, military institutions. This network of networks, formed during the 1980s, was called ARPA-INTERNET, then simply INTERNET, still supported by the Defense Department and operated by the National Science Foundation.

In 1995, motivated by commercial pressures, the growth of private corporate networks, and of non-profit, cooperative networks, the Internet was privatized. About 100,000 computer networks were interconnected around the world in 1996, with roughly 10 million computers users.[117] The internet has posted the fastest rate of penetration of any communication medium in history: in the United States, the radio took 30 years to reach 60 million people; TV reached this level of diffusion in 15

[111] http://www.arpa.mil
[112] For the full ARPANET history please refer to the bibliography 53.
[113] Source: bibliography 1, page 45.
[114] ARPA draft, III-24
[115] The first head of the Information Processing Techniques Office (IPTO), a department of ARPA.
[116] Source: Bibliography 1
[117] Source: Bibliography 54

years; the Internet did it in just three years after the development of world wide web.[118]

- **The World Wide Web**

In addition to being a means of communicating via e-mail, the Internet has become a means of propagating information via hypertext documents. Hypertext documents contain specific words, phrases, or images that are linked to other documents. A reader of a hypertext document can access these related documents as desired, usually by pointing and clicking with the mouse or using the arrow keys on the keyboard.

In this manner, a reader of hypertext documents can explore related documents or follow a train of thought from document to document, in the intertwined web of related information. When implemented on a network, the documents within such a web can reside in different machines, forming a network-wide web. Similarly, the web that has evolved on the Internet spans the entire globe and is known as World Wide Web.[119]

- **Standard Bodies**

A wide variety of organizations contribute to internetworking standards by providing forums for discussion, turning informal discussion into formal specifications, and proliferating specifications after they are standardized.

Most standards organizations create formal standards by using specific processes: organizing ideas, discussing the approach, developing draft standards, voting on all or certain aspects of the standards, and then formally releasing the completed standard to the public. They are normally independent, aimed to stimulate a healthy competition, but they are often influenced by companies and lobbies.

- **The Internet Standardization**

Some of the organizations that contributed to the foundation of Internet with internetworking standards may be found in the chapter Appendix B. The heart of Internet standardization is the IETF, which coordinate the developments in working groups. According to Scott Bradner, co-director of the Transport Area in the IETF, "IETF working groups created the routing, management, and transport standards without which the Internet would nor exist. They have defined the security standards that will help secure the Internet, the quality of service standards that will make the Internet a more predictable environment, and the standard for the next generation of the Internet protocol itself. (…) There is enough enthusiasm and expertise to make the working group a success"[120].

Apart from TCP/IP itself, all of the basic technology of the Internet was developed or refined in the IETF.

[118] Source: Bibliography 1, pg. 382
[119] Source: Bibliography 24, pg.114
[120] Bradner's full explanation about the IETF may be found in his essay on bibliography 5, pages 47-52.

Resulting from the working groups are the base documents in the standardization process, known as RFC (Request For Comment)[121]. The RFC series of documents on networking began in 1969 as part of the original ARPANET project. Initially they were submitted for comments (as the name indicates) but, after the Internet has been developed, they ave generally gone through an extensive review process before publication. In the end, they become proposals for standards (and following the standard track possibly become Draft Standards and Internet Standards), best current practice guides, informational guidelines, etc. They are freely available on the Internet and may be republished in its entirety by anyone. Interesting examples of RFC are the 2026[122], which recursively describes the Internet standards process and the 3160[123], known as “The TAO of IETF”.

The proposed standards are analysed by the IAB, which may use them to create the final standards. The research is done by the IRTF.

Also important are the IANA - responsible for names, domains and codes – and the W3C – which coordinates the work over the standards related to the Web, like the HTTP protocol and the HTML language[124].

The web standards have three different stages. The first is the draft, where the proposals are made public and are open to large-scale reviews and change. After they have been available for a certain time, the draft moves to the second stage, known as Proposed recommendation. During a 6-week period, members of the W3C vote to decide upon the adoption of the proposal as the final standard (either fully or with minor changes). If the standard is rejected, it may return to draft status for further modifications and consultation[125].

2.3.4. Trend: Open Spectrum

A radical idea, open spectrum can transform the communications landscape as profoundly as the Internet ever did, by suppressing the telephone, cable and Net access fees. The idea is that smart devices cooperating with one another function more effectively than huge proprietary communication networks, by treating the airwaves as commons, shared by all.[126]

In an open spectrum world, wireless transmitters would be as ubiquitous as microprocessors (in televisions, cars, public spaces, handheld devices) and tune themselves to free spectrum and self-assemble into networks.

If the Internet revolution started with the Arpanet, the wireless paradigm is happening via Wi-Fi, protocol that uses a narrow slice of spectrum that is already open. When spectrum licensing was established in the early 20th century, radios were primitive, as was the regulatory model used to govern them. Broadcasters needed an exclusive slice of the spectrum. Today digital technologies let many users occupy the same

[121] http://www.ietf.org/rfc.html
[122] http://www.iesg.org/rfc/rfc2026.txt
[123] http://www.iesg.org/tao.html
[124] For more information about the standards and standard bodies, see the bibliography 27.
[125] The full process is described on http://www.w3.org/Consortium/Process-20010719/
[126] Source: Bibliography 34

frequency at the same time.[127] In 2004 half of the laptops used at work are expected to have wireless connections, and in 2006 Intel hopes to incorporate transmitters into all of its processor chips.[128]

2.4. Open Trends

The current trend is the increase the level of Hardware and Software Independence, transforming IT into a commodity.

- **Applications dependent on the hardware**

Initially the customer was entirely tied to the hardware supplier. All the software was built in machine code (or in low-level languages like Assembler), different for each machine, even from the same supplier. This incompatibility brought several problems, when these machines needed to be changed or simply upgraded to a larger model:

- Compatibility – All the code should be reviewed, and sometimes completely rebuilt. Some of the code could be bought from the hardware supplier (mainly for the basic software tools), but the main part was developed internally.

- Migration tasks – The formats and protocols used to store the data were also different among the hardware platforms. To transfer the data, conversion procedures needed to be written.

- Specialization – The persons should be trained to learn the new platform. This also motivated some people to change their jobs to keep their technological knowledge and specializations.

- **Operating Systems dependent on the hardware**

The hardware suppliers started to create architectures (as the case for IBM S/360, analysed on page 21), to reduce their development tasks, cut maintenance costs, and simplify the migration tasks among the different machines using the same architecture. Nevertheless, the customer was still tied to the hardware manufacturer. To change the hardware platform, the operating system need also to be changed, implying a redesign of the application and the change of the bridges between the system and user interfaces. In most of the cases, the complete application is rebuilt, to better use the facilities from the new operating system.

As an example, this is what occurs in downsizing process, in companies migrating from mainframe applications to smaller platforms. This may take more than a decade to be fully completed.

[127] For a detailed analysis of the theory behind this, please refer to the bibliography 27, chapters 3.2 to 3.5.
[128] Source: bibliography 34

- **Hardware independence**

The development of open systems platforms – like UNIX – freed the customer from the hardware supplier. After the creation of *The Single UNIX* Specification (see page 27), the portability of the applications is paramount. However, the customer is still tied to the extensions to the base platform provided by the supplier. To change the hardware supplier or platform, the usage of the hardware extensions and software specificities should be checked and rebuilt. To avoid this, the usage of these "extra-features" should be reduced to a minimum and well documented.

Many manufacturers use the tie between their hardware and software to distinguish their systems, while other companies treat hardware and software as separate businesses.

- **Software Independence**

To reduce the efforts of maintaining different hardware platforms, while keeping the development of operating systems that satisfy different customers requirements, some companies started to standardize the hardware platforms, keeping it compatible with different operating systems. It was the case of IBM, initially within the mainframes family, one line of computers (zSeries) is compatible with three operating system families (zOs, zVM, zVSE). Recently, with IBM concentrating their revenues in the services rather than in the software or machines, all lines of IBM computers (Series Z, X, I and P[129]) are compatible with Linux. Now customers can even add Linux and Open source applications to IBM UNIX machines (pSeries), co-existing with the IBM UNIX (AIX5L) operating system.

- **Applications**

Originally, the programming process was accomplished by the arduous method of requiring the programmer to express all algorithms in the machine's language (expressed in binary digits). The first step toward removing these complexities from the programming process was the creation of assembly languages (also called second-generation languages), which replaced numeric digits by mnemonics. The evolution continued with the third-generation languages (e.g. COBOL, C), which used statements closer to the human language, traduced into the machine language by compilers.

With the development of third-generation languages, the goal of machine independence (discussed previously in this document) was largely achieved. Since the statements did not refer to the attributes of any particular machine, they could be compiled as easily for one machine as for another[130].

The evolution continued, with the implementation of software packages that allow users to customize computer software to their applications without needing hardware, operating systems, or programming languages expertise. Some of those packages

[129] Source: Bibliography 110.
[130] Reality, however, has not proven to be that simple, due to the different dialects used on different machines.

are so complex, however, that need the participation of business specialists, with an advanced knowledge of the packages, for the implementation of business functions[131].

On the chapter 3, we will se how open platforms like CORBA and Java can be used to achieve a maximum independence over the third-level languages, without the inconvenience of the software packages.

Still on the cradle is the usage of declarative programming (sometimes referred to as fifth-generation languages or logic programming), for expert systems and artificial intelligence applications.

- **The informational commodities: Hardware, Software, Applications**

In the 1960s, the computers resources were so expensive that they were shared among different companies, who bought slots of "CPU time". It is a concept known as "Time Sharing", and it started the vision of hardware as a commodity. The company does not need to buy a computer, install it, or have a special team to maintain it. Computer systems are simply used, and the fees are either fixed (based on the resources available) or based in the usage of the machines.

Nowadays, the evolution of this concept is the usage of applications as commodities. With the good performance and the relatively low price of the network communications, it's possible to run a system completely separated from the company. Moreover, with the advanced level of hardware and system independence, the only compatibility to be considered is in the application level.

That's the era of ISP (Internet Service Providers), which take care of the internet, e-mail and part of the network infrastructure, and the ASP (Application Service Providers), which supply the access to the application, freeing the company of most of the concerns related to the information and communication technologies (ICT).

Companies can run complete applications, hosted by ASPs and available via the Web. The network infrastructure can be rented from a third-party, also responsible for its maintenance. Network computers can be leased from a specialized company, which may offer the technical support. In this case, computing power is a facility at almost the same level than electricity and water.[132]

- **Open Crystal Ball**

The different hardware platforms will probably continue to broaden their initial scope, and possibly start to fusion. The division lines among them will finally disappear, each computer being able to run several different operating systems, in parallel. The open platforms are the key for this to happen, with the proprietary hardware being increasingly compliant with them, assimilating open architectures definitions.

[131] As an example, this is the case of SAP.
[132] For an analysis of the (ab)usage of services and the impact in nowadays society see the essay e*conomy, available on http://www.k-binder.be/Papers/

The end of the windows monopoly will mark the next decade. However, nobody needs another dominant operating system. The existing software platforms need to co-exist, to allow the companies and users to have choice, with the competition helping to increase the quality. The choice must be based on real business and personal needs; marketing will continue to blur the technical advantages, and to reduce this negative effect, information must be simplified and well focused to the public.

Most of the Internet tools have been developed through the open-source process. Apache – The Open source web server – still dominates the market[133]. What are the advantages of open source technology over proprietary products, for a worldwide network?

One answer is communication. The Internet is based in open standards, which were always successful to perform the connection of different hardware and software platforms. The communication was always possible and often perfect. Compatibility problems started to appear with proprietary extensions to the standards (like the HTML extensions for Netscape and Internet Explorer).

A second answer was the availability of common (and free) tools to start using the Internet and creating web pages. Proprietary solutions (like Microsoft ASP and .NET) and standards (like Shockwave) often need an expensive set of tools and machines, what limits their usage by academic and home users.

A third answer is probably “too” clear: the openness of the source code. Early web sites - using technologies like HTML and Javascript – were easily copied, altered and published again in another part of the world. This made the creation of personal web sites easy, and allowed a quick “learning by example”. The implementation of open source technologies like CGI scripts and PHP – which “hide” the source from the page – has been helped by the hacker community, which created sites with several examples of source code, which can be freely used, without royalties. The online communities maintained by commercial companies behind proprietary solutions can help to reduce this gap.

[133] According to Netcraft, in October 2002 60% of the web servers were running Apache, against 29% for Microsoft servers. To see the updated statistics, please refer to http://www.netcraft.com/survey/

3. Open Internet Development

Most of the well designed Internet applications are developed under object-oriented methodologies[134], to better cope with the modular structures of HTML and XML, and to more effectively reuse or even share programming modules with other companies. The applications using object-oriented languages are often designed using a formal methodology and tightly related to modelling languages.

In this chapter, we will analyse the development of open methodologies for application design, which can be extremely helpful to ensure scalability, security and robust execution of Web applications under stressful conditions. After, we will see three different platforms used to develop the web applications, and discuss the importance of open standards to build web services.

We cannot finish without analysing the current hype around agile development and extreme programming. To conclude, we will review some ideas about comparison elements between open and proprietary platforms.

3.1. Design

Modelling is the designing of software applications before coding, being an essential part of large software projects, and helpful to medium and small projects. Using a model, those responsible for a software development project's success can assure themselves that business functionality is complete and correct, end-user needs are met, and program design supports requirements for scalability, robustness, security, extendibility, and other characteristics, before implementation in code renders changes difficult and expensive to make.

3.1.1. MDA

The OMG Model Drive Architecture (MDA) provides an open, vendor-neutral approach to the challenge of business and technology change. Based firmly upon other open standards, MDA aims to separate business or application logic from underlying platform technology. The platform-independent models enable intellectual property to move away from technology-specific code, helping to foster application interoperability.[135]

[134] The four keys to object orientation are Encapsulation, Polymorphism, Inheritance and Instantiation
[135] Source: http://www.omg.org/mda/

- **Specification**

In September 2001, OMG members completed the series of votes that established the MDA as the base architecture for the organization's standards.[136] Every MDA standard or application is based, normatively, on a Platform-Independent Model (PIM), which represents its business functionality and behaviour very precisely but does not include technical aspects. From the PIM, MDA-enabled development tools follow OMG-standardized mappings to produce one or more Platform-Specific Models (PSM): one for each target platform that the developer chooses.

The PSM contains the same information as an implementation, but in the form of a UML model[137] instead of running code. In the next step, the tool generates the running code from the PSM, along with other necessary files. After giving the developer an opportunity to hand-tune the generated code, the tool produces a deployable final application.

MDA applications are composable: If PIMs are imported for modules, services, or other MDA applications into the development tool, it can generate calls using whatever interfaces and protocols are required, even if these run cross-platform. MDA applications are “future-proof”: When new infrastructure technologies come on the market, OMG members will generate and standardize a mapping to it, and the vendors will upgrade his MDA-enabled tool to include it. Taking advantage of these developments, cross-platform invocations can be generated to the new platform, and even port the existing MDA applications to it, automatically using the existing PIMs. Although MDA can target every platform and will map to all with significant market buy-in, CORBA[138] plays a key role as a target platform because of its programming language-, operating system-, and vendor-independence. The mapping from a PIM to CORBA has already been adopted as an OMG standard.[139]

[136] The initial definition of the MDA is set out in the document "Model Driven Architecture - A Technical Perspective", by the OMG Architecture Board MDA Drafting Team.
[137] See chapter 3.1.3 on page 57
[138] See chapter 3.1.2 on page 55
[139] Source: Bibliography 97

- **The core**

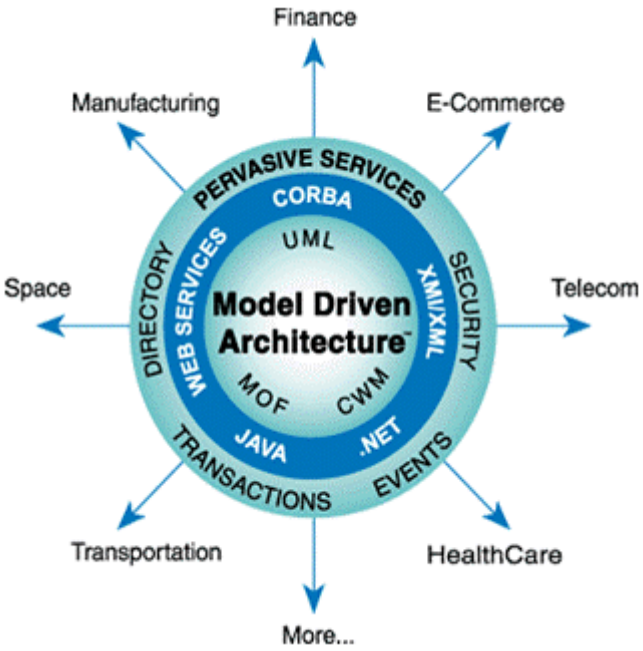


Figure 16 – Model Driven Architecture[140]

- The Unified Modeling Language (UML)

As discussed, each MDA specification will have, as its normative base, two levels of models: a Platform-Independent Model (PIM), and one or more Platform-Specific Models (PSM). These will be defined in UML, making OMG's standard modeling language the foundation of the MDA. UML is discussed separated in the next chapter.

- The Meta-Object Facility (MOF)

By defining the common meta-model for all of OMG's modeling specifications, the MOF allows derived specifications to work together in a natural way. The MOF also defines a standard repository for meta-models and, therefore, models (since a meta-model is just a special case of a model).

- Common Warehouse MetaModel (CWM)

The CWM standardizes a complete, comprehensive metamodel that enables data mining across database boundaries at an enterprise and goes well beyond. Like a UML profile but in data space instead of application space, it forms the MDA mapping to database schemas. The product of a cooperative effort between OMG and the Meta-Data Coalition (MDC), the CWM does for data modelling what UML does for application modelling.

[140] Source: http://www.omg.org/mda/

3.1.2. CORBA[141]

CORBA is the acronym for Common Object Request Broker Architecture, OMG's open, vendor-independent architecture and infrastructure that computer applications use to work together over networks. Using the standard protocol IIOP, a CORBA-based program from any vendor, on almost any computer, operating system, programming language, and network, can interoperate with a CORBA-based program from the same or another vendor, on almost any other computer, operating system, programming language, and network. CORBA is useful in many situations because of the easy way that CORBA integrates desktop and servers from so many vendors.

- **Technical overview**

In CORBA, client and object may be written in different programming languages. CORBA applications are composed of objects[142]. For each object type, an interface is defined in OMG IDL (Interface Definition Language)[143]. The interface is the syntax part of the contract that the server object offers to the clients that invoke it. Any client that wants to invoke an operation on the object must use this IDL interface to specify the operation it wants to perform, and to order the arguments that it sends. When the invocation reaches the target object, the same interface definition is used there to parse the arguments and perform the requested operation.

This separation of interface from implementation is the essence of CORBA. The interface to each object is defined very strictly. In contrast, the implementation of an object - its running code, and its data - is hidden from the rest of the system (that is, encapsulated). Clients access objects only through their advertised interface, invoking only those operations that that the object exposes through its IDL interface, with only those parameters (input and output) that are included in the invocation.

[141] Source: Bibliography 100.
[142] Individual units of running software that combine functionality and data, and that frequently (but not always) represent something in the real world.
[143] The IDL interface definition is independent of programming language, but maps to all of the popular programming languages via OMG standards: For example, OMG has standardized mappings from IDL to C, C++, Java, COBOL, Smalltalk, Ada, Lisp, Python, and IDLscript.. See http://www.omg.org/gettingstarted/omg_idl.htm for more details.

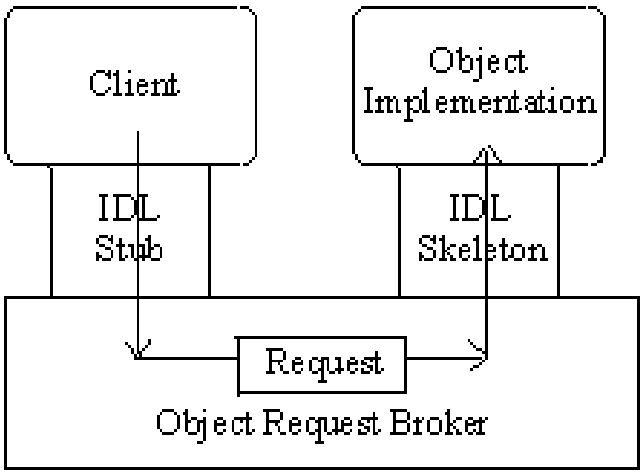


Figure 17 – CORBA request from client to object[144]

The above figure shows how everything fits together, at least within a single process: The IDL is compiled into client stubs and object skeletons[145]. Passing through the stub on the client side, the invocation continues through the ORB (Object Request Broker)[146], and the skeleton on the implementation side, to get to the object where it is executed. Because IDL defines interfaces so strictly, the stub on the client side matches perfectly with the skeleton on the server side, even if the two are compiled into different programming languages, or even running on different ORBs from different vendors.

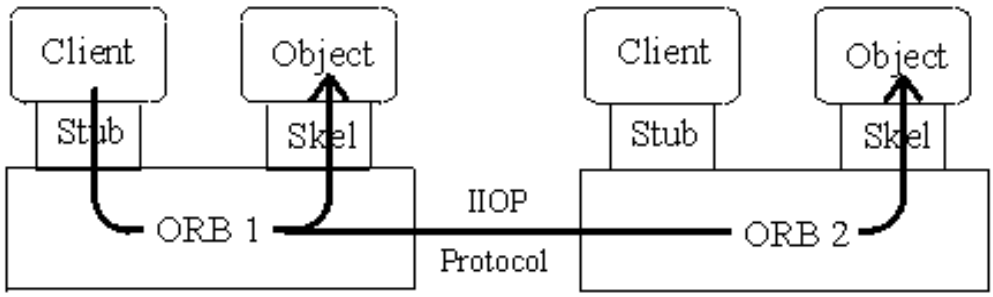


Figure 18 – CORBA remote invocation flow using ORB-to-ORB communication[147]

The figure above diagrams a remote invocation. In order to invoke the remote object instance, the client first obtains its object reference. To make the remote invocation, the client uses the same code that it used in the local invocation we just described, substituting the object reference for the remote instance. When the ORB examines the object reference and discovers that the target object is remote, it routes the invocation out over the network to the remote object's ORB. Although the ORB can tell from the object reference that the target object is remote, the client cannot. This ensures location transparency - the CORBA principle that simplifies the design of distributed object computing applications.

[144] © 2000 OMG
[145] Stubs and skeletons serve as proxies for clients and servers, respectively.
[146] For more details on ORB see http://www.omg.org/gettingstarted/orb_basics.htm
[147] © 2000 OMG

3.1.3. UML

The Unified Modelling Language (UML) is a open method for specifying, visualizing, constructing, and documenting the artefacts of software systems, as well as for business modelling and other non-software systems. It represents a collection of the best engineering practices that have proven successful in the modelling of large and complex systems, addressing the needs of user and scientific communities.[148]

- **Medieval ages**

Identifiable object-oriented modelling languages began to appear between mid-1970 and the late 1980s[149] as various methodologists experimented with different approaches to object-oriented analysis and design. Several other techniques influenced these languages, including Entity-Relationship modelling and the Specification & Description Language. These early methods could not satisfy most of the design requirements, and started to incorporate each other's techniques. A few clearly prominent methods emerged, including the OOSE, OMT-2, and Booch'93 methods. Each of these was a complete method, and was recognized as having certain strengths. In simple terms, OOSE was a use-case oriented approach that provided excellent support business engineering and requirements analysis. OMT-2 was especially expressive for analysis and data-intensive information systems. Booch'93 was particularly expressive during design and construction phases of projects and popular for engineering-intensive applications. [150]

- **All for one**

The UML started out as collaboration among three outstanding methodologists: Grady Booch (Boock'93), Ivar Jacobson (OOSE), and James Rumbaugh (OMT-2). At first Booch and Rumbaugh sought to unify their methods with the Unified Method v. 0.8 in 1995; a year later Jacobson joined them to collaborate on the slightly less ambitious task of unifying their modelling languages with UML 0.9.

The user community quickly recognized the advantages of a common modelling language that could be used to visualize, specify, construct and document the artefacts of a software system. They enthusiastically applied early drafts of the language to diverse domains ranging from finance and health to telecommunications and aerospace. Driven by strong user demand, the modelling tool vendors soon included UML support in their products. [151]

[148] Source: Bibliography 12
[149] The number of identified modelling languages increased from less than 10 to more than 50 during the period between 1989-1994
[150] Source: Bibliography 12
[151] Source: Bibliography 98

- **One for all**

At the same time that UML was becoming a de facto industry standard, an international team of modelling experts assumed the responsibility to make the language a formal standard as well. The “UML Partners”, representing a diverse mix of vendors and system integrators, began working with the three methodologists in 1996 to propose UML as the standard modelling language for the OMG. The Partners organized themselves into a software development team that followed a disciplined process. Since the process was based on an iterative and incremental development life cycle, the team produced frequent “builds” and draft releases of the specification.

The Partners tendered their initial UML proposal to the OMG (UML 1.0) in January 1997. After nine months of intensive improvements to the specification, they submitted their final proposal (UML 1.1) in September 1997, which the OMG officially adopted as its object-modelling standard in November 1997. UML 1.5 is the current specification adopted by the OMG, and in mid-2001, OMG members started work on a major upgrade to UML 2.0. Four separate RFPs - for UML Infrastructure, UML Superstructure, Object Constraint Language, and UML Diagram Interchange - keep the effort organized.

The OMG defines object management as software development that models the real world through representation of "objects." These objects are the encapsulation of the attributes, relationships and methods of software identifiable program components. A key benefit of an object-oriented system is its ability to expand in functionality by extending existing components and adding new objects to the system. Object management results in faster application development, easier maintenance, enormous scalability and reusable software.[152]

OMG members are preparing to standardize a Human-Usable Textual Notation (HUTN) for UML models, or at least those that fit into the Enterprise Distributed Object Computing (EDOC) UML Profile.

- **(Almost) Ten Years Later**

UML defines twelve types of diagrams, divided into three categories:

- Structural Diagrams represent static application structure and include the Class Diagram, Object Diagram, Component Diagram, and Deployment Diagram.

- Behavior Diagrams represent different aspects of the application’s dynamic behavior and include the Use Case Diagram (used by some methodologies during requirements gathering); Sequence Diagram, Activity Diagram, Collaboration Diagram, and Statechart Diagram.

- Model Management Diagrams represent ways the applications modules can be organized and managed and include Packages, Subsystems, and Models.

The Advanced UML Features add to the expressiveness of UML:

[152] Source : http://www.omg.org/news/about/index.htm

- Object Constraint Language (OCL) has been part of UML since the beginning and express conditions on an invocation in a formally defined way: invariants, preconditions, post conditions, whether an object reference is allowed to be null, and some other restrictions using OCL. The MDA relies on OCL to add a necessary level of detail to PIMs and PSMs.

- Action Semantics UML Extensions are a recent addition and express actions as UML objects. An Action object may take a set of inputs and transform it into a set of outputs, or may change the state of the system, or both. Actions may be chained, with one Action's outputs being another Action's inputs. Actions are assumed to occur independently - that is, there is infinite concurrency in the system, unless you chain them or specify this in another way. This concurrency model is a natural fit to the distributed execution environment of Internet applications.

UML Profiles tailor the language to particular areas of computing or particular platforms. In the MDA, both PIMs and PSMs will be defined using UML profiles; eventually OMG will define a suite of profiles that span the entire scope of MDA. Examples of three supporting UML Profiles and one specialized profile are[153]:

- The UML Profile for CORBA defines the mapping from a PIM to a CORBA-specific PSM.

- The UML Profile for EDOC is used to build PIMs of enterprise applications. It defines representations for entities, events, process, relationships, patterns, and Enterprise Collaboration Architecture.

- The UML Profile for EAI defines a profile for loosely coupled systems[154]. These modes are typically used in Enterprise Application Integration, but are used elsewhere as well.

- A UML Profile for Schedulability, performance, and time supports precise modelling of predictable systems, precisely enough to enable quantitative analysis.

- **Opening the iron mask**

UML is clearly an open methodology, and the three main characteristics are the independency of the technical infrastructure, the independency of the methodology and the openness of its definition, managed by a recognised and independent organisation (OMG).

At first, UML can be used to model any type of application, running on different combinations of hardware and software platforms, programming languages and network protocols. Built upon the MOF metamodel which defines class and operation as fundamental concepts, UML is a natural fit for object-oriented languages and

[153] Source: Bibliography 99
[154] Loosely coupled are those systems that communicate using either asynchronous or messaging-based methods

environments like C++, C#, Java and Python. Some UML tools[155] analyse existing source code and reverse-engineer it into a series of UML diagrams. Other tools execute UML models in interpretative way (to validate the design) and other may even generate program language code from UML.

The process of gathering and analysing an application's requirements, and incorporating them into a program design, is a complex one and the industry currently supports many methodologies that define formal procedures specifying how to go about it. The second characteristic of UML is that it is methodology-independent. Regardless of the methodology used to perform the analysis and design, UML can express the results. Using XMI (XML Metadata Interchange, another OMG standard), the UML model can be transferred from one tool into a repository, or into another tool for refinement or the next step in the chosen development process.[156]

At last, the UML definitions are openly discussed by an independent organisation – OMG, freely published in the Web and implemented by any software vendor, including several open source projects.

The OMG is structured into three major bodies, the Platform Technology Committee (PTC), the Domain Technology Committee (DTC) and the Architecture Board. The consistency and technical integrity of work produced in the PTC and DTC is managed by an overarching Architectural Board. Within the Technology Committees and Architectural Board rest all of the Task Forces, SIGs, and Working Groups that drive the technology adoption process of the OMG.

There are three major methods of influencing the OMG process, in addition to the impact of general review, commentary and open discussion. The first is the ability to vote on work items or adoptions in the Task Forces that are ultimately reviewed and voted on at the Technology Committee level. The second is the ability to vote on work items or adoptions at one or both of the Technology Committee levels. The third is the ability to actually submit technology for adoption at one or both of the Technology Committee levels. Membership fees are based on these levels of influence.[157]

[155] A series of UML tools can be found on http://www.objectsbydesign.com/tools/umltools_byCompany.html
[156] Source : Bibliography 97
[157] Source : http://www.omg.org/news/about/index.htm

3.2. Web Platforms

3.2.1. Java

The Java platform is based on the power of networks. Since its initial commercial release in 1995, Java technology has grown in popularity and usage because of its true portability. Java is a general-purpose concurrent object-oriented programming language. Its syntax is similar to C and C++, but it omits many of the features that make C and C++ complex, confusing, and unsafe. Java was initially developed to address the problems of building software for networked consumer devices. It was designed to support multiple host architectures and to allow secure delivery of software components. To meet these requirements, compiled Java code had to survive transport across networks, operate on any client, and assure the client that it was safe to run.

- **Java platform**

A platform is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms in the chapter 2. Most platforms can be described as a combination of the operating system and hardware. However, Java is a software-only platform that runs on top of other hardware-based platforms.

The Java platform was designed to run programs securely on networks and allows the same Java application to run on many different kinds of computers. For example, PersonalJava applications power home appliances, Java Card applications run on smart cards, smart rings, and other devices with limited memory, and Java TV applications run in television settop boxes. This interoperability is guaranteed by a component of the platform called the Java virtual machine (or "JVM") – a kind of interpreter that turns general Java platform instructions into tailored commands that make the devices do their work.[158]

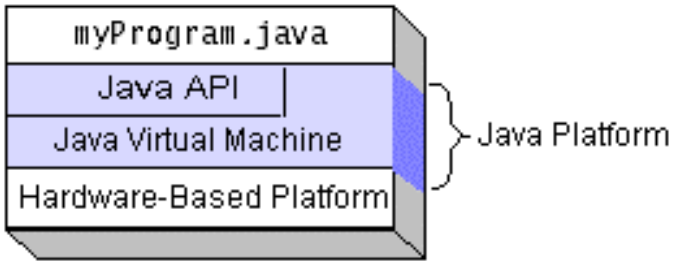


Figure 19 – Java platform components[159]

The other component – Java API – is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages.

[158] Source: http://java.sun.com/java2/whatis/
[159] Source: http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html

The power of compiled languages is the execution speed. The power of interpreted languages is the flexibility to run a same program in different platforms. The Java programming language is unusually powerful in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Java bytecodes —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java bytecode instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed.

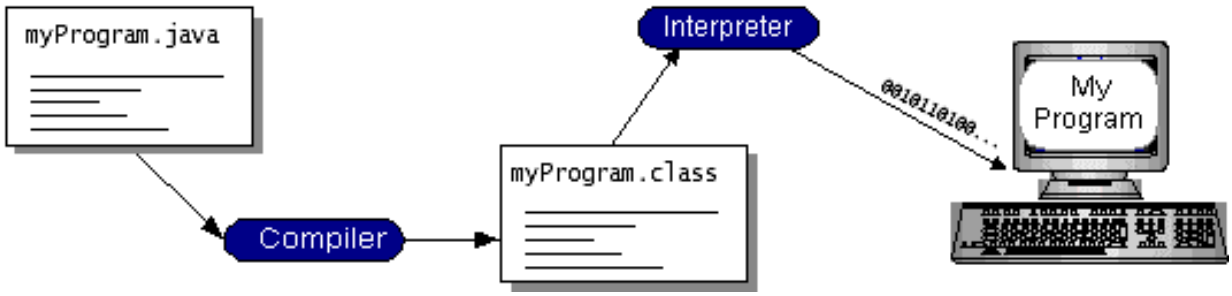


Figure 20 – Java compiler and interpreter[160]

- **Java editions**

Three editions group the different technologies according to the hardware platform: J2ME (tiny commodities, like smartcards), J2SE (development of applets and applications) and J2EE (enterprise server-side applications).

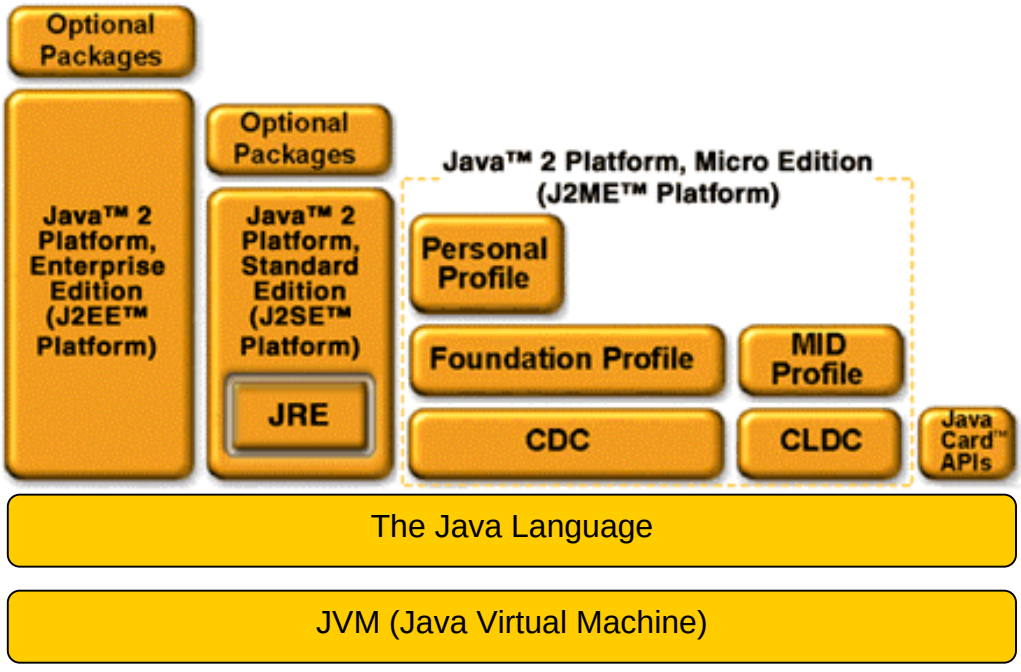


Figure 21 – The Java platform and editions[161]

[160] Source : http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html
[161] Source: http://java.sun.com/java2/whatis/

Java is owned by Sun Microsystems, which has in place the appropriate mechanisms to meet the evolving needs of the industry, and to fulfil the needs of Internet application developers[162].

- **Java buzzwords[163]**

In the scope of this document, let us analyze the two main Java characteristics[164] in detail, and briefly describe the others:

- Architecture neutral

The independence of the architecture is fundamental to allow the same program to run virtually anywhere, without special adaptation tasks. For Java, this is guaranteed by the Java platform components seen above. This is seen as a menace by companies like Microsoft, which tried to block the compatibility of the JVM with recent Windows versions. This was a strategy to force the Windows customers to use the Microsoft platform - .NET – and luckily has been avoided by the justice.

- Portable

Portability is closely related to the architecture independence. It describes the characteristics of the language that must remain unchanged independently of the platform, like data types (e.g. character, byte, integer). Even if this may seem simple at first sight, this is not true for most of the languages, in which one of the activities when porting a program from one platform to another is the adaptation of the data types. Java did not make this mistake. The data types have specific, defined lengths regardless of the system.

The Java virtual machine is based primarily on the POSIX interface specification – an industry-standard definition of a portable system interface. Implementing the Java virtual machine on new architectures is a relatively straightforward task.

- Simple – Reduces software development cost and time to delivery
- Object oriented – To function within increasingly complex, network-based environments, programming systems must adopt object-oriented concepts
- High performance – Obtained from the compiled bytecode
- Interpreted – The bytecode is transformed into executable code during the execution of the program
- Multithreaded – Multiple tasks can be executed in parallel, improving overall system performance
- Robust – Provides a solid, reliable environment in which to develop software
- Dynamic - Classes are linked only as needed. New code modules can be linked in on demand from a variety of sources, even from sources across a network.

[162] Source: bibliography 13
[163] Source : http://java.sun.com/docs/white/langenv/
[164] Source: http://java.sun.com/docs/white/langenv/Intro.doc2.html#379

- **Applets**[165]

Sun's HotJava browser showcases Java's interesting properties by making it possible to embed Java programs inside HTML pages. These programs, known as applets, are transparently downloaded into the HotJava browser along with the HTML pages in which they appear. Before being accepted by the browser, applets are carefully checked to make sure they are safe. Like HTML pages, compiled Java programs are network and platform-independent. Applets behave the same way regardless of where they come from, or what kind of machine they are being loaded into and run on.

With Java as the extension language, a Web browser is no longer limited to a fixed set of capabilities. Programmers can write an applet once and it will run on any machine, anywhere. Visitors to Java-powered Web pages can use content found there with confidence that it will not damage their machine.

- **Open Belgian Cathedrals**

According to Tony Mary[166], openness and flexibility are the main Java advantages. This openness could be widened further, with a creation of a unique media technology by a group of companies. This could be compared – accorded to him – to the cathedrals. In opposition to the medieval castles – strongly protected and closed, and almost destroyed today – the Cathedrals always remained open and survived.

According to Edy Van Asch[167], Open Source is the main Java trend. Java has always been closely related to Open Source, but since 2001 this relationship became stronger. Examples are Eclipse (Open Source Java Development environment), Junit (Open Source Test Structures), Jboss (Open Source Application Server) and Jini.

Jini network technology[168] is an open architecture that enables developers to create network-centric services – whether implemented in hardware or software – that are highly adaptive to change. Jini technology can be used to build adaptive networks that are scalable, evolvable and flexible as typically required in dynamic computing environments. Jini Offers an open development environment for creative collaboration through the Jini Community, and is available free of charge with an evergreen license. It extends the Java programming model to the network (by moving data and executables via a Java object over a network) and enables network self-healing and self-configuration.

[165] Source : http://java.sun.com/docs/books/vmspec/html/Introduction.doc.html
[166] New director of VRT (Belgian public television), in an interview to Datanews n°37, 29/11/2003 (Page 1)
[167] Profession Leader Technology Engineering at CGE&Y, in an interview to Datanews n°37, 29/11/2003 (Page 10)
[168] Source : http://wwws.sun.com/software/jini/

3.2.2. .NET

.NET (read as “dot-net”) is both a business strategy from Microsoft and its collection of proprietary programming support for the Web services[169]. Its goal is to provide individual and business users with a seamlessly interoperable and Web-enabled interface for applications and computing devices and to make computing activities increasingly Web browser-oriented. The .NET platform includes servers (running Microsoft Windows), building-block services (such as Web-based data storage), device software and Passport (Microsoft's identity verification service). Functionally the .Net architecture can be compared to the Java platform. In the scope of our study, they are completely different: Java aims to give the companies and users the liberty of choosing the hardware and software platforms. .NET is tightly related to Microsoft Windows operating systems (the only software platform supported) and favours the usage of complementary Microsoft products such as Internet explorer and the Office suite. Omnipresence is the goal, and the weapon is the relative complexity of Java platform.

The full release of .NET is expected to take several years to complete, with intermittent releases of products such as a personal security service and new versions of Windows and Office that implement the .NET strategy coming on the market separately.

- **Dot objectives[170]**

According to Bill Gates, Microsoft expects that .NET will have as significant an effect on the computing world as the introduction of Windows. One concern being voiced is that although .NET's services will be accessible through any browser, they are likely to function more fully on products designed to work with .NET code.

- The ability to make the entire range of computing devices work together and to have user information automatically updated and synchronized on all of them
- Increased interactive capability for Web sites, enabled by greater use of XML rather than HTML
- A premium online subscription service, that will feature customized access and delivery of products and services to the user from a central starting point for the management of various applications, such as e-mail, for example, or software, such as Office .NET
- Centralized data storage, which will increase efficiency and ease of access to information, as well as synchronization of information among users and devices
- The ability to integrate various communications media, such as e-mail, faxes, and telephones
- For developers, the ability to create reusable modules, which should increase productivity and reduce the number of programming errors

[169] See chapter 3.3 on page 72 for more info about web services
[170] Source: Bibliography 92

- **Dot components**

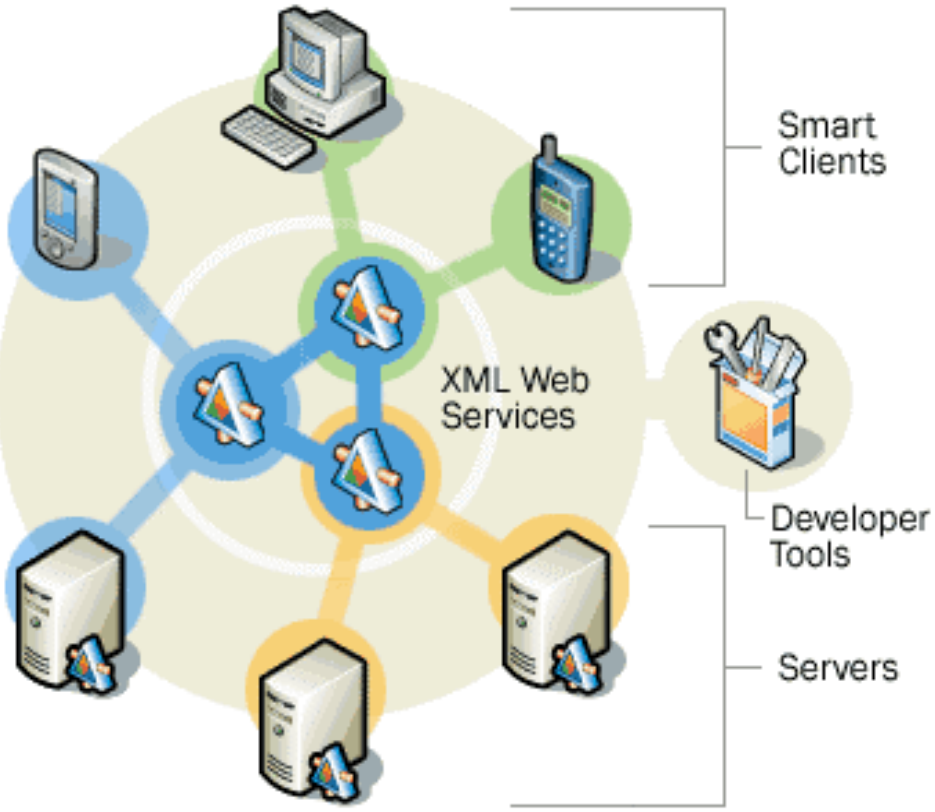


Figure 22 - The Components of Microsoft .NET-Connected Software[171]

- Smart clients: “Smart” is the term used by Microsoft for client computers using one of its proprietary operating systems: Microsoft Windows XP, Windows XP Embedded, and Windows CE .NET. All other clients platforms are excluded from the .NET architecture.

- XML Web services let applications share data, and invoke capabilities from other applications without regard to how those applications were built, what operating system or platform they run on, and what devices are used to access them. While XML Web services remain independent of each other, they can loosely link themselves into a collaborating group that performs a particular task.

- Developer Tools – Microsoft Visual Studio .NET (with high-level programming languages like Visual Basic, Visual C++ and C#) and the Microsoft .NET Framework (The application execution environment, which include components like the common language runtime, a set of class libraries for building XML Web services, and Microsoft ASP.NET) aim to supply a complete solution for developers to build, deploy, and run XML Web services.

- Servers – In opposite to the portability of Java applications, only servers running Microsoft operating systems (Windows 2000 Server, Windows Server 2003, and the .NET Enterprise Servers) are certified to fully obtain good results from the .NET platform.

[171] Source: Bibliography 90

- **Dot integration**

Microsoft considers XML "deceptively simple"[172], and would certainly prefer a complex and proprietary solution to integrate the .NET platform – composed by Microsoft operating systems and basic software – with other vendors environments. XML is revolutionizing how applications talk to other applications — or more broadly, how computers talk to other computers — by providing a universal data format that lets data be easily adapted or transformed. Therefore, Microsoft saw in XML an opportunity to connect its completely proprietary platform with other proprietary or open environments. As an example, XML is extensively used by Microsoft Host Integration Server 2000 to provide application, data, and network integration between .NET platforms and host systems (Mainframe and AS/400 environments, considered "legacy" by Microsoft, and estimated to contain 70 percent of all corporate data).

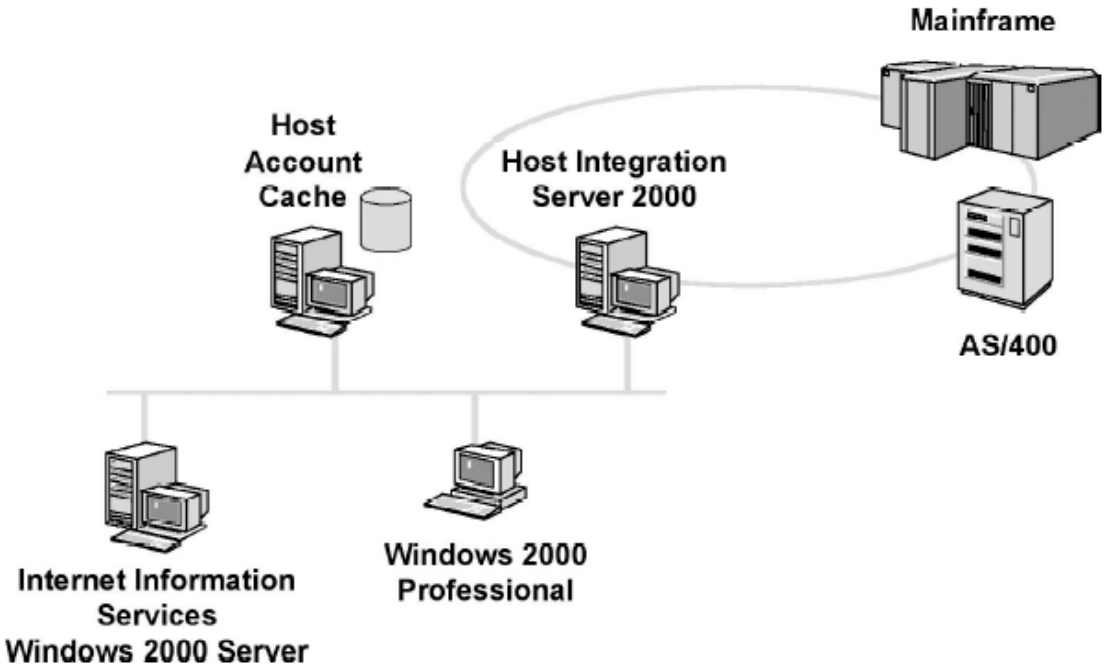


Figure 23 - Microsoft Host Integration Server[173]

[172] Source : Bibliography 87
[173] Source: Bibbliography 93

3.2.3. Java.NET

The comparison between Java and .net is not only technical. Economical and cultural aspects are far more important. In this chapter, we will analyse some of the differences and exploit possible alternatives.

- **Co-existence**

Considering the easy integration of both .NET and Java platforms, by the usage of XML and Web services, many analysts suggest a co-existence of both platforms in most companies. Scott Dietzen[174] foresee this to happen in the next five years. One of his arguments is the current .NET power in the client-side and J2EE in the server-side. But he also prophesises that the integration of Java to the companies culture, and its implementation should become easier, before Microsoft .NET technology is mature enough to attract large customers in projects where reliability is too important. Nico Duerinck[175] agrees with the co-existence to be possible, mainly if Web services are implemented, even if a reduced number of companies are currently using both technologies.

- **Portability**

For new projects, Java may be clearly preferred to ensure the portability of the applications. It guarantees that a server may start small, running Windows operating systems, and move to other platforms like Linux or UNIX to satisfy more advanced requirements like parallelism, high-availability and reliability. An application developed under .NET will always require a windows-based server to be able to exploit all its capabilities.

Additionally, as described by Dietzen, the Java community is composed by more than 100 vendors, working to improve the Java features and developing complementary and concurrent solutions. The innovation resulted is far beyond what a single company can afford.

- **Symbiosis**

According to David Chappell[176], Windows DNA (seed of .net) and Java appeared together in 1996. Since then, J2EE was built using windows DNA interesting concepts, and .net framework incorporated Java-like features. This “mutual and cross-pollenization” helps innovation in both senses and minimize the problem of “single source of creativity”, suggested by Dietzen.

- **Infrastructure Costs**

According to Duerinck, most of the companies will try to keep the existing infrastructures, and will tend to implement .net platforms in windows systems and

[174] Scott Dietzen, BEA Systems CTO, interview to Datanews n°37, 29/11/2002, Page 6
[175] Nico Duerinck, EDS, interview to Datanews n°37, 29/11/2002, Page 10
[176] David Chappell, independent ;Net specialist, interview to Datanews n°08, 28/02/2003, Page 3

Java for all the other architectures. It is often a simple financial decision. If a Microsoft license exists, it will probably be kept and exploited. The cost of .net framework in existing windows systems is very low. Some Java application servers can be rather expensive, however cheap and open source alternatives exist.

The training costs and simplification of the hardware and operating systems diversity is also an important point to be considered.

- **Complementary products**

Products are often developed by independent companies to be compatible with both platforms. This is the case of Compuware development products[177] and Real Software development architecture[178]. This can also minimize the costs to transport an application from one platform to another, by reducing the effects of .net proprietary philosophy.

3.2.4. The outsiders: LAMP

The term LAMP (Linux, Apache, MySQL, Perl / Python / PHP) refers to a set of Open Source software tools that allow for the development and deployment of web applications[179].

All of the components of LAMP can be downloaded without any costs. The support is done by a network of developers aggregated in online communities like "OnLamp[180]" to help each other get the most out of LAMP. Training is available from a wide array of providers, and many consulting firms offer advanced capabilities for those businesses that require sophisticated LAMP development.

I have used LAMP to develop my web site[181]. All dynamic pages have been developed in PHP, and the databases in MySQL. The website is hosted by ALL2ALL[182] on Linux and Apache servers. The usage of Open Source allows the web hosting to be performed with low costs. The learning cost was a book about

[177] Source: Datanews n°08, 28/02/2003, Page 8
[178] Source: Datanews n°37, 29/11/2002, Pages 2-3
[179] http://www.onlamp.com/pub/a/onlamp/2001/01/25/lamp.html
[180] http://www.onlamp.com/
[181] www.k-binder.be
[182] www.all2all.org

PHP and MySQL[183] complemented by a large amount of free online information, from FAQS to complete systematic training. The learning time was reduced by using the large amount of online examples. Most of the sites using LAMP provide the sources used for their development.

- Linux

See chapter 2.2.3 on page 30 for a large discussion on Linux.

- Apache[184]

The most widely used web server software in the world[185], Apache can be run on a variety of operating systems, including AIX, Digital UNIX, FreeBSD, Irix, Mac OS X, Netware, OpenBSD, Solaris, SunOS, Windows, and of course, Linux. Apache's security record is far better than that of Microsoft IIS.

- MySQL[186]

This database was built with the philosophy that a web database should be lean and fast. It doesn't incorporate the diverse array of application server features that Oracle and Microsoft database tools do, instead focusing on core performance and leaving enhanced functions to scripting languages. MySQL runs on a variety of operating systems, including AIX, FreeBSD, Mac OS X, Solaris, Windows, and Linux. It is well known for its reliability; while the recent "Slammer" worm that crippled much of the Internet was spread through Microsoft SQL Server, it did not affect MySQL databases.

- Perl[187]

This scripting language was invented to help overworked system administrators. It still performs that function admirably well, but is also often used in web development for things like dynamic forms, server monitoring, and database integration. Perl runs on a wide variety of operating systems.

- PHP[188]

Developed from the ground up as a Web developer's language, PHP is easy to use and executes very quickly. Skilled developers can use PHP to build everything from online forms to complex database-driven web applications. PHP is extremely popular, runs on a variety of operating systems, and is most often used in conjunction with MySQL.

One of the most powerful PHP features is the extensive library of functions, open to be expanded by volunteer work. It contains functions to address any possible need, to communicate with a large range of databases, and to read and write files in many different formats, like PDF. The documentation is written by volunteer work and

[183] See bibliography 11
[184] http://www.apache.org
[185] To see the updated statistics, please refer to http://www.netcraft.com/survey/
[186] http://www.mysql.com
[187] http://www.perl.org
[188] http://www.php.net

consists in examples supplied by the community. For any developer trained in other languages the learning by example process is extremely efficient.

- Python[189]

A general-use high-level language, Python is often used to tie different backend pieces together. It lends itself particularly well to Java/XML integration and development of dynamic content frameworks. Python runs on several operating systems.

- Trends

To give a large impulse to this initiative, Sun unveiled a dual-processor server - LX50 – that runs the new Sun distribution of the Linux OS. The server will run Intel x86 chips and will be available not only with Linux but also with an Intel-enabled version of Sun's own Solaris operating system. In a briefing with analysts and press, Sun's new executive vice president of software, Jonathan Schwartz, further detailed the vendor's Linux strategy. While Sun will keep pushing Solaris in the data centre and J2EE for middle-tier business logic, it is making a major bet on LAMP as the application environment of choice at the edge of today's corporate infrastructures[190].

A company called BD-X used LAMP as the basis for its web services strategy. BD-X's chief technology officer Kevin Jarnot started prototyping a wide-area content-creation environment that would allow BD-X to provide XML-based financial content on demand for a wide variety of subscription customers. "At first, our decision to use Open Source was to build a prototype without a lot of cost," according to Jarnot. But after his prototype was built, Jarnot found, "it would actually be fine for [the] project's required performance and scalability."

BD-X founders reasoned that if they could transcribe these calls, mark them up into XML, and then offer them to business customers, they might just have a viable business. In the end, Jarnot says that 60 percent of the reason BD-X ended up with an Open Source solution was because of cost. Thirty percent, he says, was because "we were doing some pretty bleeding edge stuff... if we had been using proprietary software, it would have been a lot harder to do the integration." Oh, and the last ten percent of why BD-X went with Open Source? "Coolness factor," says Jarnot.[191]

Jarnot said he expected BD-X would eventually need to port the system to a more production ready commercial environment like Solaris running Macromedia's ColdFusion. But, Jarnot said, it turned out that his Open Source prototyped version was good enough to take into live production. Using the standard Open Source LAMP stack as a backend platform, they also decided to use the Open Source ELVIN message-oriented middleware to filter the transcriptions and deliver them to customers. On the XML side, the developers used Jedit (Open Source Java text editor) for XML markup and editing. Jarnot mentions quite a few options he is considering for using J2EE and Microsoft .NET technologies in conjunction with his Linux, PHP and Apache systems. What proves again the clear advantages of the symbiosis theory, between commercial, proprietary and open platforms.

[189] http://www.python.org
[190] Source: http://www.internetwk.com/webDev/INW20020813S0008?ls=TW_090302_fea
[191] Source: Bibliography 95

3.3. Web Services

According to Brad Murphy[192], "by now, virtually every company is fully aware and engaged in various stages of integrating the Internet into their business model to help serve customers better. A key part of every integration strategy is the use of Web services. The phenomenal growth of Web services is made possible by open standards that are built into the products that comprise the Web services infrastructure".

With a standards based infrastructure, companies have the ability to use the Web rather than your own computer for various services, linking applications and conducting B2B or B2C commerce regardless of the computing platforms and programming languages involved. In a way, Web services are the natural evolution of the Internet – again all based on open standards. At this point, Web services may seem straightforward, but a critical issue companies face is choosing the best enterprise/strategic development platform for creating and deploying these new interoperable applications.

- **Open standards**

Several options are available, but the key thing to remember is that any Web services development platform – by definition – must follow open standards to be interoperable, be innovative with rich and robust functionality and work properly in responding to and addressing the complex needs of a typical enterprise organization. The W3C has working groups focused on refining the SOAP 1.1 and WSDL 1.1 specifications, which should improve things considerably. The XML Protocol Working Group is working on SOAP 1.2 while the Web Services Description Working Group is creating the WSDL 1.2 specification. Meanwhile, the IETF and OASIS are also working on standardizing Web service specifications, including DIME and WS-Security.

While work at the W3C focuses on new versions of the core Web services specifications, a separate organization is focusing more attention on interoperability. The Web Services Interoperability Organization (WS-I) is focused on defining best practices for ensuring Web service interoperability. The WS-I Basic Profile Working Group is currently developing a set of recommendations for how to use the core Web service protocols like SOAP 1.1 and WSDL 1.1 to maximize interoperability.

The clear leaders in the race to deliver Web services functionality are IBM and the entire Java / J2EE vendor community. They recognized nearly five years ago the need to support standards that provide customers the flexibility of choice and as a result, devoted serious resources to creating its solution. As discussed on chapter 3.2.2, the Microsoft approach is very different when it comes to implementation. IBM and the Java community are committed to a Web infrastructure based on open standards – especially Java and Linux – to make disparate systems work together. Microsoft presents .Net as an open platform that supports Web standards, but it is still a highly proprietary technology that runs solely on the Windows platform. Microsoft supports Web standards interoperability – such as XML and SOAP – but only within the proprietary Windows framework that is not easily portable.

[192] Source: Bibliography 85

- **Proprietary platforms**

Locking-in Web services to the Windows platform may be a perfectly acceptable solution for a small or mid-tier business. However, for larger organizations, this approach is not only impractical but also caries with it risks and hidden costs when addressing the integration and collaboration challenges of significant internal applications as well as a large, dispersed organization. It's important to consider that almost every large company has a rich, complex heterogeneous computing environment.

Another critical factor to consider is that a commitment to only one operating environment such as Windows means companies lose the ability to choose and negotiate among Web service application vendors for the best function and price. Will companies really want to marry their Web services application development – including the possibility of a substantial portion of their future Internet-based revenue and vital customer relation activities – so closely to Microsoft?

Microsoft's commitment to standards and interoperability in the Web services world is both welcome and critical to large-scale adoption in the marketplace. The fact that Microsoft has embraced standards and interoperability with the Java world is evidence that Microsoft recognizes customers will no longer tolerate proprietary solutions. It's also true that the .NET offerings will likely capture a large segment of the existing pool of Microsoft developers and customers. Unfortunately for larger customers however, this won't produce increased platform choice or vendor flexibility.

If future Web usage, cost savings and revenue generation is only half of what is projected, it is likely that most companies will continue their investment and support of Java as their strategic platform. Will .NET gain acceptance and find an important role to play in a Java/Standards-based world? Absolutely, but only if Microsoft delivers on their promise to truly support standards. Which raises an interesting question – does history give anyone reason to believe that Microsoft is truly interested in customer choice and flexibility based on vendor neutral standards? Ignorance is bliss.

- **Open source**[193]

Not only big names (IBM, Microsoft, BEA) are behind web services software. Open Source software and tools exist such as the Apache Tomcat servlet engine[194], the JBoss J2EE-based server[195] and Apache AXIS (a Java toolkit for building and deploying Web service clients and servers)[196].

According to Thomas Murphy, "Consider the fact that Web services themselves are built on top of Open Source technologies like HTTP and TCP/IP, people don't realize the amount of Open Source technology that underlies everything that they do. Many companies use Open Source technology areas other than Web services without giving it a second thought, and so should open themselves to using the technology for Web services as well. Look at the Internet — the most predominant server is Apache, which is Open Source, and so people don't really have problems running

[193] Source: Bibliography 94
[194] More info on http://jakarta.apache.org/tomcat/
[195] More info on http://www.jboss.org
[196] More info on http://xml.apache.org/axis

Open Source software. And look at the underlying protocols; they're all Open Source as well."[197]

Those involved in the Open Source community, not surprisingly, tout the benefits of Open Source tools for Web services development. Marc Fleury, founder and president of JBoss says that many commercial Web service tools are "pricy implementations rushed to the market with poor quality." He believes that because of this, most commercial vendors will disappear over time, but that Open Source technology will survive because of its superior quality. Additionally, he adds, Web services technology is a "moving target. Many implementations are fighting for standard status. Going with a free software implementation guarantees you the maximum probability of going with a standard." It's not only the Open Source community and analysts who believe that Open Source technologies are the best solution to Web services development — those involved in Web services creation and deployment are backers as well. For example, FiveSight Technologies[198] provides comprehensive Web services workflow integration and software tools, and they've built those tools using Open Source software.

Paul Brown, president of FiveSight, says that "Without using Open Source, we wouldn't have been able to launch our company. If FiveSight on its own, or any other company, had to implement XML schema or WSDL or any other number of Web services technologies in combination, it would be an impossible task. The Open Source community is a catalyst for innovation in software, and so I know things like where we can get a good Open Source implementation of a transaction manager. It's an opportunity to solve a hard problem by building on work from the community at large. We've used Open Source in our development work, from the first piece of software we deployed. We didn't have the money to pay for developers and staff," and so instead turned to Open Source software, which already had the software available, to do the job.

Murphy also notes the drawbacks. When companies devote themselves to using Open Source technology for Web services, they're taking on responsibility for product support and management, since there is no commercial vendor that takes care of that for them. That means no technical support, and no clear upgrade paths. There may also be legal issues involved with Open Source licenses, and so businesses need to have their legal staffs examine the implications of using Open Source before committing.

Another issue is that the Open Source community has yet to fully embrace Web services technologies with open arms. JBoss's Fleury, for example, says that "Web services isn't real so far. We see zero dollars in Web services." There are signs, however, that that is changing and an increasing number of Open Source tools and developers have turned their attention to Web services.

[197] Thomas Murphy, senior program director for Meta Group consulting firm
[198] www.fivesight.com

3.4. Agile Development

Agile Development[199] is a collection of methods and practices aimed to reduce the time and effort required to develop an application, through communication, simplicity, feedback, courage and humility. Some of its processes are Adaptive Software Development[200], Crystal[201], Scrum[202], Xbreed[203], Dynamic Systems Development Method (DSDM)[204] and Usage-Centered Design (UCD)[205]. However, the best known agile method is XP (Extreme Programming).

3.5. Extreme Programming

In the early 1990s, a man named Kent Beck was thinking about better ways to develop software. Kent worked together with Ward Cunningham to define an approach to software development that made every thing seem simple and more efficient. Kent contemplated on what made software simple to create and what made it difficult. In March of 1996, Kent started a project using new concepts in software development: the result was the Extreme Programming (XP) methodology. Kent defined Communication (programmers communicate with their customers and fellow programmers), Simplicity (the design is simple and clean), Feedback (quick feedback by testing the software starting on day one), and Courage (the programmers are able to courageously respond to changing requirements and technology) as the four values sought out by XP programmers.

Extreme Programming (XP) is actually a deliberate and disciplined approach to software development, and it's successful because it stresses customer satisfaction. The methodology is designed to deliver the software required by the customer in the target timeframe, and empowers the developers to confidently respond to changing customer requirements, even late in the life cycle. This methodology also emphasizes teamwork. Managers, customers, and developers are all part of a team dedicated to delivering quality software. XP implements a simple, yet effective way to enable groupware style development.

XP is an important new methodology for two reasons. It is a re-examination of the software development practices used as standard operating procedures, and it is one of several new lightweight software methodologies created to reduce the cost of software. XP goes one step further and defines a process that is simple and enjoyable.[206]

[199] See http://www.agilelogic.com/resources.html
[200] See http://www.adaptivesd.com/
[201] See http://www.crystalmethodologies.org/
[202] See http://www.controlchaos.com/
[203] See http://www.xbreed.net/
[204] See http://www.dsdm.org/index.asp
[205] See http://www.foruse.com/
[206] Source: http://www.extremeprogramming.org/

According to Kent, software development is a difficult task and the efforts should be concentrated in the four main activities:[207]

- Listening – Consists in obtaining information from clients, users, managers, and business people. The problem must be identified and the data must be collected for testing purposes.

- Designing – Some contradictory opinions exist around this topic. Some think that XP consists in low design, short-sighted. As this is far from a preferred option, let us consider the other opinion stream: the design must be performed by the developer and be validated by the user, but the time spent shall not be extremely high.

- Coding – The heart of development.

- Testing – Validate the application, with the help from the customer in the final phases.

3.5.1. Extreme Programming and Open Source[208]

- **XP Open Values**

The four XP values fit with the hacker's ethic:

- Most open source projects rely on communications. As normally the teams are geographically dispersed, the exchange of written messages (via e-mail, newsgroups and forums) may have a positive effect as it allows a record of the discussions and the comment from the community.

- Simplicity is often present – and sometimes even exaggerated – in open source projects, which are normally started to satisfy basic needs of one person or one part of the community.

- The feedback is one of the most important features of open source development, as the responses from the community can be based on the functionality (new features required, conceptual and technical problems) but are often recommendations on the programming techniques (via the sharing of the source code) or suggested new routines, functions or updates.

- It normally takes a lot of courage to start a new open source project, as the source code may be analysed and criticised by a huge number of peer programmers, but also to join an ongoing project and commit to using the spare time to perform the coding.

- **XP Open Practices**

- Planning - XP assumes you have a customer (or a proxy) as part of the project. Open Source projects generally don't have a well defined "customer". Nor is there

[207] Kent Beck, quoted by Wikipedia.org.
[208] This chapter has been freely adapted from the bibliography 101

a single voice for their users. Instead, Open source projects tend to be guided by a combination of the vision of their key developers, and the "votes" of their other developers and users.

- Small releases - Release early, release often. This is a current practice in XP and Open Source projects

- Testing - Automated tests are a key part of XP, as they give the courage for re-factoring and collective ownership. They are what allow the small releases (release early, release often XP and Open Source principle) to happen. Although many Open Source projects include test suites, this is possibly the best XP practice to be adopted. An Open Source project that incorporated XP style tests would have enormous advantages. It would require less oversight and review and it would encourage more people to contribute because they would have immediate feedback on whether their changes worked or not. Moreover, it would ensure that as the software changed, grew and ported, it wouldn't break.

- Re-factoring – “Re-factoring is the process of changing a software system in such a way that it does not alter the external behaviour of the code yet improves its internal structure."[209] Many Open Source projects are reviewed and re-factored, but often reluctantly. In XP re-factoring is recommended all the time, so that the design stays as clean as possible. The fear, of course, is that something will be broken, but this is counteracted by having automated tests, and pair (or peer) programming.

- Pair programming – One of XP recommendations is to always have two persons working in the same programs. This is rather difficult to achieve in the real world of commercial development, but compensated by the Open Source peer programming, with an exponential number of programmers revising important projects.

- Collective ownership - In XP collective ownership means that anyone can change any part of the project – programmers are not restricted to a certain area of expertise. Open Source projects normally work differently, having restrictions on the persons allowed to make changes directly, and the process required for submitting changes. This is necessary because of the large numbers of contributors. However, in the Open Source software all the parts of the software are visible to every programmer, who can always request access to change “restricted” parts of the code in special cases.

- Continuous integration - Some Open Source projects integrate contributions continuously and others group them into releases. Some projects allow anyone to commit changes whereas others require changes to be submitted for review. Nevertheless, on an individual level, most hackers do practice continuous integration.

- 40-hour week - It’s hard to apply this to hackers that are working in their spare time. The main objective of this XP principle, however, is satisfied by the Open Source definition: motivation.

[209] Quoted from “Martin Fowler, Refactoring” : www.refactoring.com

- On-site customer - Most Open Source projects don't have a physical "site" or a specific "customer". However, the essence of this practice is to have good communications between programmers and users, and this has already been explained.

- Coding standards - Most Open Source projects adhere to coding standards, for the same reasons that XP requires them: To allow different programmers being responsible for the same code at different times. On smaller projects, the standards are often informally enforced by the key developer, who adapts the contributions into his own style. Large projects usually have an explicit set of standards.

- **Open Source XP software**

- JUnit[210] is a regression testing framework written by Erich Gamma and Kent Beck. It is used by the developer who implements unit tests in Java. JUnit is Open Source Software, released under the IBM's Common Public License Version 1.0 and hosted on SourceForge

- CurlUnit[211] is a an instance of the xUnit architecture commonly used as a testing framework within the Extreme Programming methodology (XP) methodology. Like Junit, CurlUnit is Open Source Software, released under the IBM Public License and hosted on SourceForge.

Other resources for Open Source development under XP rules may be found in http://www.xprogramming.com/software.htm.

- **Final word**

Open source software can be used in Extreme Programming development, and XP concepts can be adopted by Open Source projects. The design part should be given more importance, tough, with the integration of XP, Open Source and MDA. The goal s being to reduce the time and increase the quality of open source development, and to reduce costs and improve efficiency and portability of applications developed under XP principles.

3.6. Open decision[212]

A good starting point in deciding whether to use Open Source software is to take a look at what phase a company is, regarding the development of Web services, says Thomas Murphy, senior program director for Meta Group consulting firm: "Open Source is ideal for when you need to keep up with where technology is headed, and for that portion of your technical staff working in future technologies. It's well suited for these early stages. But when you're looking to use something for production and development, that requires a stable release path. So I think that often, people look for commercial tools at that point."

[210] See http://www.junit.org/index.htm
[211] See http://curlunit.sourceforge.net/
[212] Source: Bibliography 94

Another deciding factor is money. Open Source may be the perfect solution for companies that haven't decided yet to go full-bore into Web services, or those that are operating on a shoestring or looking to hold down costs,.

An exceedingly important issue, but one that is easy to overlook is the "cultural" factor, says Eric Promislow, senior developer specializing in Web services with ActiveState, which provides Open Source-based applications, tools and support. "Go with Open Source for Web services development if your staff already uses Open Source for other purposes, and want to stay with the tools they know and love," he says. "They know how it works, and they know how to get support for it from the Open Source community." They'll be far more productive — and happier in their jobs — than if they had to use commercial tools.

Promislow adds that Open Source software is an ideal way for companies to dip their toes in the water when it comes to Web services development, and so is suited for companies still deciding whether to seriously pursue Web services. "They can make no upfront investment except in time," he says, and so can inexpensively develop pilot projects.

Murphy warns that there are a series of not-obvious issues that companies should be aware of when they ultimately decide to go the Open Source route. "Businesses need to understand that in using Open Source, they are also taking on support and product management responsibilities," he warns. "You need someone to track where the bug fixes are, and know what the newest features are likely to be. It's not like a commercial product where someone makes money off support and so provides it for you — you have to do it yourself." Of course the existence of service companies around Open Source is increasing and reducing the effect of this problem.

Another costly activity may be installing the software. Not all Open Source software have easy installation routines, and often a commercial implementation of Open Source software is required to ensure the community support, the low cost, and technical support.

Open Source software is built by community, and is a community effort, and so businesses have to decide up-front how involved their developers can be in that community. Should companies allow developers not only to work with Open Source software, but also participate in the community, and in the community development of the software? Will managers allow developers to spend company time working on Open Source community projects? The rules shall be put in place for this ahead of time, to avoid problems in later project phases.

4. Common structures for data exchange

4.1. Content

A basic concept of computer sciences – explained in the very first class of any good introduction course[213] – is that any information is stored in a computer by using the binary digits (bits). Only zeroes and ones, and there must be basic rules to translate them to alphabetic and numeric symbols, comprehensible by the human race and - for the multimedia environments - to transform them into formatted text, images, sound, etc. This is the realm of codes and formats.

4.1.1. Character Codes

When the first ideas about networking appeared, data was not standardized. Each proprietary system used a different form of data representation, considering only the needs for local applications and, in the best cases, the local languages. Some standardization works started, and different standards have been implemented in parallel by computer manufacturers and independent organizations. The most important ones (in terms of current adoption) are briefly explained here.

- **EBCDIC**

EBCDIC[214] is a binary code for alphabetic and numeric characters that IBM developed in the 1960s together with the first mainframe architecture, the System/360. This code is still used to represent the information in all IBM mainframe platforms, with each alphabetic or numeric character represented with an 8-bit binary number. 256 possible characters (letters of the alphabet, numerals, and special characters) can be defined, and different tables exist for country-specific characters.

It is a code elaborated to fit programming needs. As it is intimately related to the architecture, the IBM Assembler language could use the logical distribution of the numbers and letters in EBCDIC to make validations, translations and arithmetical operations.

- **ASCII**

ASCII (American Standard Code for Information Interchange) was developed in 1963 by the American National Standards Institute (ANSI[215]). It is the most common format for text files in computers and is the American National Version of ISO/IEC 646. Each character is represented with a 7-bit binary number (a string of seven 0s or 1s). 128 possible characters are defined.

[213] Source: Bibliography 24
[214] EBCDIC is pronounced either "ehb-suh-dik" or "ehb-kuh-dik" and stands for Extended Binary Coded Decimal Interchange Code.
[215] ANSI is presented on page 128

UNIX and DOS-based operating systems use ASCII for text files. Even IBM's PC and workstation operating systems use ASCII instead of IBM's proprietary EBCDIC. Conversion programs allow different operating systems to change a file from one code to another.

- **ISO**

There is a joint technical subcommittee of ISO and IEC to deal with information technology to promote the standardization of graphic character sets and their characteristics, associated control functions, their coded representation for information interchange and code extension techniques. It is identified as ISO/IEC JTC1/SC2, which published several standards[216].

- **Unicode**

In 1991, the ISO Working Group responsible for ISO/IEC 10646 and the Unicode Consortium decided to create one universal standard for coding multilingual text. Since then, they have worked together very closely to extend the standard and to keep their respective versions synchronized.

Officially called the “Unicode Worldwide Character Standard”, Unicode[217] is a system for the interchange, processing, and display of the written texts of the diverse languages of the modern world, also supporting many classical and historical texts in a number of languages. Currently, the Unicode standard contains 34,168 distinct coded characters derived from 24 supported language scripts. These characters cover the principal written languages of the world and additional work is underway to add the few modern languages not yet included. Although the character codes are synchronized between Unicode and ISO/IEC 10646, the Unicode Standard imposes additional constraints on implementations to ensure that they treat characters uniformly across platforms and applications. To this end, it supplies an extensive set of functional character specifications, character data, algorithms and substantial background material that is not in ISO/IEC 10646.

For each character defined in Unicode there is an assigned code point: a hexadecimal number that is used to represent that character in computer data.

4.1.2. Multimedia

Several open and proprietary formats have been defined for the exchange of multimedia documents. To discuss them is out of the scope of this document, and more references may be found on the Internet:

Audio: http://www.diffuse.org/audio.html
Video: http://www.diffuse.org/video.html
Images: http://www.diffuse.org/raster.html

[216] A list can be found on :
http://www.iso.ch/iso/en/stdsdevelopment/tc/tclist/TechnicalCommitteeStandardsListPage.TechnicalCommitteeStandardsList?COMMID=23
[217] http://www.unicode.org/

4.1.3. Formats for the Document Interchange[218]

Paper documents are linear, normally read in the order specified by the author. Hypertext documents, on the other hand, are structured in a non-structured way, to allow the information to be obtained in a sequence determined by the areas of interest of the reader. The hypertext structures are composed by nodes (the documents) and links (the references linking two different nodes, or two different segments of the same node)[219]. The evolution of the hypertext documents is briefly explained here:

- **SGML**

SGML provides an object-oriented method for describing documents (and other information objects with appropriate characteristics). The standard defines a set of semantics for describing document structures, and an abstract syntax of formally coding document type definitions. Apart from defining a default (concrete) syntax, based the ISO 646 code set, that can be used for text and markup identification when no alternative is specified, SGML does not suggest any particular way in which documents should be structured but allows users to define the structure they require for document capture or presentation.

SGML has made its principal impact in markets making use of structured textual information. This has particularly included those markets managing and producing technical documentation, although not exclusively so.

Its take up elsewhere has steadily increased, especially following the arrival of the World Wide Web, where it has been used as the formal basis for HTML and XML.

- **HTML**

HTML is the data format that has made the World Wide Web possible. Its first proposal has been written by Tim Berners-Lee (member of CERN in Switzerland) and described a mark-up language that was able to execute in a heterogeneous distributed environment. HTML documents are SGML documents with generic semantics that are appropriate for representing information from a wide range of domains. HTML markup can represent hypertext news, mail, documentation, and hypermedia; menus of options; database query results; simple structured documents with in-lined graphics; and hypertext views of existing bodies of information.

HTML has been in use by the World Wide Web (WWW) global information initiative since 1990, when the MOSAIC program has been created by the University of Illinois, based on Berners-Lee's proposal. Version 2.0 (RFC 1866) roughly corresponds to the capabilities of HTML in common use prior to June 1994.

An extended version (4.0) of the HTML specification was released to the public on 8th July 1997 and became an approved W3C Recommendation on 18th December

[218] http://www.diffuse.org/docs.html

[219] For more information, see bibliography 27. For an anthropological view of the hypermedia environment, see the excellent study from Pierre Levy (bibliography 8 – page 45 and bibliography 9 – page 200).

1997. The extensions include facilities for multilingual data presentation, interactive elements and objects and control of presentation using cascading style sheets.

- **XHTML**

The Extensible Hypertext Markup Language (XHTML™) is a family of current and future document types and modules that reproduce, subset, and extend HTML, reformulated in XML. XHTML Family document types are all XML-based, and ultimately are designed to work in conjunction with XML-based user agents. XHTML is the successor of HTML, and a series of specifications has been developed for XHTML. But … what is XML?

4.2. XML

4.2.1. Introduction

- **Metadata**

Giovinazzo states, "The universal delivery of information, both within the corporation as well as to its partners, requires system and device independence. The need, therefore is to devise a common language for the communication of data while maintaining the structure and context of that data. The solution is a metalanguage, a language whose primary function is to express the metadata surrounding data. (…) Metadata is data about data. It defines for us the structure, format and characteristics of the data. Typically, metalanguages are referred to as mark-up languages, but (…) the mark-up aspects of a metalanguage are just one of the main characteristics"[220].

Metadata is used to give a meaning to the data, transforming it into information. This is the main difference of XML if compared to HTML. HTML contains simply the data, together with formatting instructions. XML allows the metadata to be sent together with the data, giving a meaning to the information transmitted via the network[221]. The information can then be interpreted by automatic processes, stored in databases, displayed and sent to other recipients. The requirement for all the processes is the ability to recognize the XML format and convert the information into other formats[222] used to store and display information. This allows the information to easily flow across a peer-to-peer network of companies and individuals[223], independently of the types of software involved in the communication and data processing.

- ***Fiat Lux***

The World-Wide-Web consortium (W3C)[224] began discussions in 1996 to define a mark-up language with the power and extensibility of SGML but the simplicity of HTML. In February 1998 the version 1.0 of the XML specification has been approved.

[220] In bibliography 13
[221] Since identifying the data gives you some sense of what means (how to interpret it, what you should do with it), XML is sometimes described as a mechanism for specifying the semantics (meaning) of the data.
[222] As discussed in the section 4.1on page 80.
[223] This will be discussed in the section 5 on page 90
[224] See Appendix B (Standards Organizations) on page 128.

One of the first real-world applications was Microsoft CDF (Channel Definition Format).[225]

- **Objectives**

The objectives of XML recognize SGML's complexity and structure as well as HTML's simplicity and lack of structure. XML is not a replacement for HTML or SGML but a complement to them. The three basic XML objectives are:

- Extensibility - the tags are not fixed and controlled by the standard organizations like HTML, but defined by document authors, which allow the creation of language extensions, shared by many nodes in a network.
- Structure - XML makes it possible to support structures like hierarchies and data associations, and to divide a document into its components and parts.
- Validation - a valid document strictly complies with the mark-up and syntax of a particular network environment.[226]

- **Usages**

XML can be used in different ways:

- Traditional data processing – XML encodes the data for a program to process.
- Document-driven programming – XML documents are containers that build interfaces and applications from existing components.
- Archiving – The foundation for document-driven programming, where the customized version of a component is saved (archived) so it can be used later
- Binding – The DTD or schema that defines an XML data structure is used to automatically generate a significant portion of the application that will eventually process that data

4.2.2. Technical Strengths[227]

- Format Independence – Changes to display do not depend on changing the data. A separate style sheet specifies the display format.
- Portability – because the display is "extracted" from the data, this becomes portable. Therefore, the code is often shorter and interoperable.
- Searching – searching the data is both easy and efficient. Search engines can simply parse the tags rather than the raw data, becoming "intelligent".

[225] Source: Bibliography 29, page 4
[226] Source: Bibliography 13, chapter 9
[227] Source: Bibliography 29, page 6

- Collaboration – In conjunction with Internet applications, XML's associated linking facilities make possible for many persons to work in the same document.

- Repository – XML is the emerging standard for repositories, which are becoming the primary means for storing and relating software system components.

- Relationships – XML allows the exchange of communication containing complex relationships like trees and inheritance.

- Self-describing code – this is a self-describing item.

4.2.3. Openness

XML complement the technologies discussed previously in this document. It is a open standard by excellence, and can be associated with Open Source programs, operating systems and internet technologies to obtain a maximum openness and independence.

- Plain Text – Since XML is not a binary format, the files can be created or edited with standard text editors or visual development environments. That makes it easy to debug programs, and makes it useful for storing small amounts of data. At the other end of the spectrum, an XML front end to a database makes it possible to efficiently store large amounts of XML data as well. Therefore, XML provides scalability for anything from small configuration files to a company-wide data repository[228].

- Wide open standard – In a similar way than Open Source – where both the executable and source codes are available, allowing any programmer to understand the detailed processes and information exchange between the programs – XML opens the access to the different layers of information: Data, description and display format. By certifying that all the information is stored and exchanged in XML format, it is guaranteed that it will always be accessible, independently of the supplier of the software used for its processing.

- XML and Open Source – If Open Source programs are used to process the information stored in XML, the complete open environment is available, and a complete independence of the supplier may be guaranteed. Future changes in the direction of technology may imply in some programs or formats to become obsolete. In this case, the information, its formats and all the processing algorithms may be easily understood and rewritten.

- Security and Privacy – When the description is available together with the data, the reasons to hold some kinds of information may be understood. When XML is associated with Open Source, the lack of privacy – by sending secret information to hidden recipients – may be discovered. Companies and governmental agencies may be certified of the information to be exchanged with the network. This does not happen with the usage of proprietary – and close – code and

[228] Source: Bibliography 84

formats, as many cases of privacy and security breaches have been found recently.

4.2.4. XML components

XML allows the design of new, custom-built languages. Before a draft of the new XML language appears, designers must agree on three things: which tags will be allowed, how tagged elements may nest within one another and how they should be processed. The first two – the language's vocabulary and structure – are typically codified in a Document Type Definition, or DTD. The XML standard does not compel language designers to use DTDs, but most new languages will probably have them, because they make it much easier for programmers to write software that understands the mark-up and does intelligent things with it. Programmers will also need a set of guidelines that describe, in human language, what all the tags mean.

Schemas, like DTDs, define the structure and semantics of an XML document, but in a more verbose way, using XML to define the rules and allowing for a richer set of data types to do so.

Publishers – who would often like to "write once and publish everywhere" – may extract the substance of a publication and then show it in different forms, both printed and electronic. XML allows this to happen by tagging content to describe its meaning, independent of the display medium. Publishers can then apply rules organized into "stylesheets" to reformat the work automatically for various devices. The standard for XML stylesheets is called the Extensible Stylesheet Language, or XSL.

4.2.5. Industry Applications

XML is being put across industry platforms. Groups of interest create working tasks to identify the information types used in specific domains, to document the data structures and to codify a DTD, creating a new language. The wide-range of applications that are exploiting the XML standard provide some indication of the widespread interest in the language. For example:

- cXML (Commerce XML) – Developed in conjunction with more than 40 companies, it is a set of lightweight XML DTDs, based on XML, with their associate request/response process.

- OTP (Open Trading Protocol) – It provides an interoperable framework for Internet commerce. It is able to handle cases where the shopping site, the payment handler, the delivery handler and the support provider are performed by different parties or by one party.

- XML/EDI – The integration for XML and EDI[229] (Electronic Data Interchange) is a logical step for electronic commerce. XML/EDI provides a standard format to describe different types of data (e.g. a loan application, an invoice, an healthcare

[229] EDI works by providing a collection of standard message format and element dictionary in a simple way for businesses to exchange data via any electronic messaging service.

claim) so that the information can be decoded, manipulated and displayed consistently and correctly.

- MathML (Mathematical Mark-up Language) – A XML application for describing mathematical notations and capturing both their structure and content. Its goal is to enable mathematics to be served, received and processed on the Web.

4.3. Trends

The expanding development of Frameworks is probably going to expand to all main commercial and academic areas. As an example, according to Bosak and Bray[230], "From the outset, part of the XML project has been to create a sister standard for metadata. The Resource Description Framework (RDF), finished on February 2003, should do for Web data what catalogue cards do for library books. Deployed across the Web, RDF metadata will make retrieval far faster and more accurate than it is now. Because the Web has no librarians and every Webmaster wants, above all else, to be found, we expect that RDF will achieve a typically astonishing Internet growth rate once its power becomes apparent." [231]

It is virtually possible to reorganize any existing structure, from the web or the technical infrastructure, by using XML. One example is the development of a standard for XML-based hypertext, named XLink and due later this year from the W3C. It will allow the user to choose from a list of multiple destinations. Other kinds of hyperlinks will insert text or images ad hoc, instead of forcing you to leave the page. Bosak and Bray argue "XLink will enable authors to use indirect links that point to entries in some central database rather than to the linked pages themselves. When a page's address changes, the author will be able to update all the links that point to it by editing just one database record. This should help eliminate the familiar '404 File Not Found' error that signals a broken hyperlink." [232]

To conclude by quoting again Bosak and Bray, "The combination of more efficient processing, more accurate searching and more flexible linking will revolutionize the structure of the Web and make possible completely new ways of accessing information. Users will find this new Web faster, more powerful and more useful than the Web of today. (…) Web site designers, on the other hand, will find it more demanding. Battalions of programmers will be needed to exploit new XML languages to their fullest. And although the day of the self-trained Web hacker is not yet over, the species is endangered. Tomorrow's Web designers will need to be versed not just in the production of words and graphics but also in the construction of multilayered, interdependent systems of DTDs, data trees, hyperlink structures, metadata and stylesheets--a more robust infrastructure for the Web's second generation." [233]

[230] Jon Bosak and Tim Bray played crucial roles in the development of XML. Bosak, an on-line information technology architect at Sun Microsystems in Mountain View, California, organized and led the World Wide Web Consortium working group that created XML. He is currently chair of the W3C XML Coordination Group and a representative to the Organization for the Advancement of Structured Information Standards. Bray is co-editor of the XML 1.0 specification and the related Namespaces in XML and serves as co-chair of the W3C XML Syntax Working Group.
[231] Source: Bibliography 96
[232] Source: Bibliography 96
[233] Source: Bibliography 96

Of course the peer-to-peer networks, the collaborative works and the powerful aggregation of hackers may find innovative ways of learning and may adapt themselves to this new way of constructing the Internet. Possibly, as we are going to discuss in the next chapter, the hackers are better equipped than the institutions to adapt easier and quicker to new demanding technologies. The key word is motivation.

Part II – Brave Open World

"I don't know the future. I Didn't come here to tell you how this is going to end. I came here to tell you how it's going to begin" Neo in *The Matrix*

5. The Network Society

5.1. Networks

"It is not proper to think of networks as connecting computers. Rather, they connect people using computers to mediate. The great success of the Internet is not technical, but in human impact. Electronic mail may not be a wonderful advance in Computer Science, but it is a whole new way for people to communicate. The continued growth of the Internet is a technical challenge to all of us, but we must never loose sight of where we came from, the great change we have worked on the larger computer community, and the great potential we have for future change." – David Clark

Licklider, as David Clark, was among the first to notice the social importance of networks, and to envision the future paradigm of a global network. It is important that the technology evolved as we analysed in the previous chapter, providing the infrastructure that allowed this network to be possible. However, the computers, network devices and protocols were simply the media to get the communication performed. Similar to the psychophysical definition of the sound, as a wave transmission that is perceived by the ear, the information available in the network only becomes interesting when it's consulted by a user. The users of a network are the elements that count, as stated by Metcalfe: "The community value of a network grows as the square of the number of its users increase"[234]. The mission of people like Metcalfe was to be like the first architects that built cities and planned squares, working to give the material conditions for the people to meet and communicate.

We saw that the first networking concepts were built around the definition of common protocols, open standards and the open systems. After the networks were established, they have been used to improve the communication among the researchers, and revealed to be the best way to exchange information, discuss ideas, have agreements, publish them, and restart the cycle. This created a positive spiral, which elevated the performance of the network[235]:

- With an increasing *consistency*, due to the high sharing of knowledge between the users, aimed to increase the efficiency and usage of the network.

- With a high *connectedness*, due to the increasing efficiency and usage of the network, and to its reduced cost and complexity.

We may also note two external factors with participated in the factors above:

- The incentive of the governments – and most specifically the military organizations during the cold war – that funded the infrastructure and sponsored the academic institutions. This allowed the research to produce the knowledge and be connected to the network.

[234] Source: Metcalfe's law. Metcalfe being the creator of the Ethernet - one of the basic technical elements of the Internet infrastructure – his original sentence was possibly something like "the power of the network increases exponentially by the number of computers connected to it".
[235] Measured according to Castells' criteria, defined on bibliography 1, page 187. This is tightly related to the second part of Metcalf's sentence: "Therefore, every computer added to the network both uses it as a resource while adding resources in a spiral of increasing value and choice".

- The increasing performance of the microchips at a given price, as specified in the Moore's law[236], which implied in reducing costs of computer power and hardware components, allowing more users to be connected to the network.

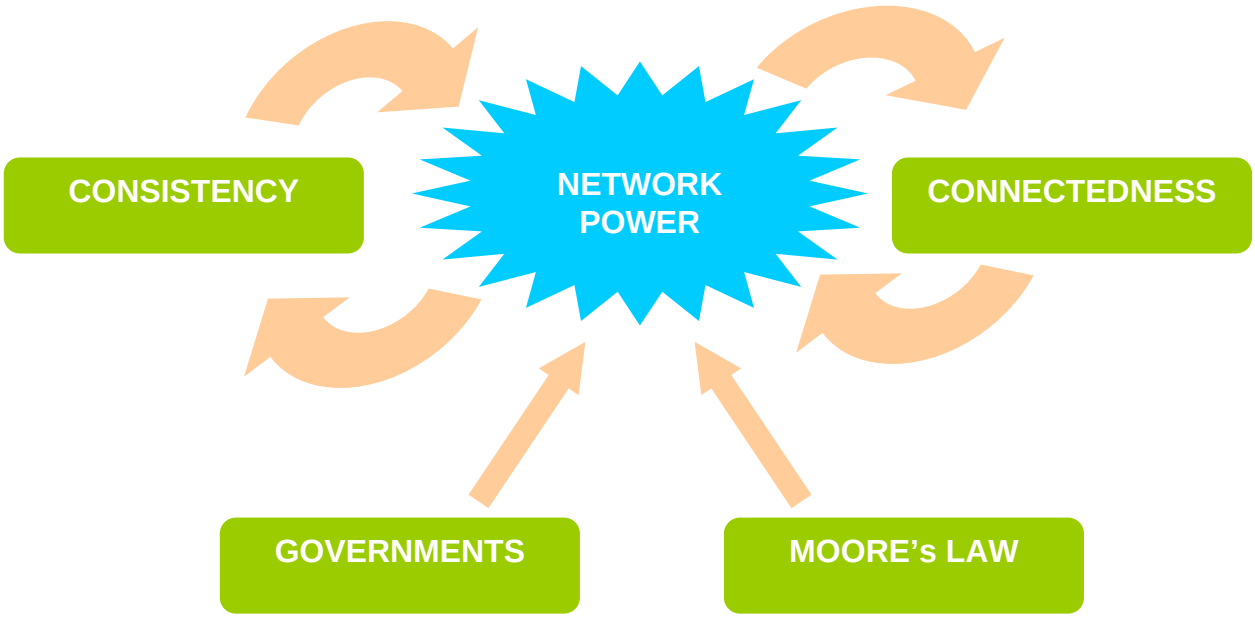


Figure 24 – Network power spiral and external factors

5.2. The Network Enterprise

5.2.1. From merchant networks

Initially, the networks were used by high-tech companies – like ATT's Bell labs, Rand Corporation, BBN (Bolt, Beranek & Newman), IBM – to exchange information with the scientific community, and to progressively build stronger infrastructure devices, protocols and software. IBM still uses its revolutionary proprietary protocols and standards to promote a cohesive work among the different R&D centres all over the world, smoothly integrate them with the other departments and – by the usage of standard interfaces, TCP/IP and OSI-compliant protocols – with the academic institutions. IBM understood it should increase its sources from all forms of knowledge, to remain an innovative firm. Innovation was critical and companies like Sun and 3COM have flourished in this environment. The computer scientists participating of this networked environment alternated or cumulated jobs in the industry and in the academic institutions. This created a networked milieu of innovation whose dynamics and goals became largely autonomous from the specific initial purposes[237].

[236] An accepted "law" in the electronics industry. See the original article in bibliography 57.
[237] Source: Bibliography 1

5.2.2. To the merchantable network

Later all the enterprises started to be connected, via the network, with suppliers, customers, service providers and research laboratories over the world, in a multicultural framework, forming network enterprises, and founding the global economy. Thus, the information economy emerged in a planetary level, in different cultural/national contexts, evolving around a common matrix of organisational form in the processes of production, consumption, and distribution[238].

Castells identified an important shift from vertical bureaucracies (the hierarchical oligopolies from the industrial era) to the horizontal corporation (the networked companies that survived and thrived in the informational economy), "dynamic and strategically planned network of self-programmed, self-directed units based on decentralization, participation, and coordination. (…) The manner in which a company shares information and systems is a critical element in the strength of its relationships" [239].

The networks formed by the horizontal corporations are divided, according to Ernst[240], into intra-firm (link different divisions and business functions from the company) and inter-firm (normally relying suppliers, producers or customers). However, the scope of our study is to identify two other important inter-firm connections, also identified by Ernst:

- Standards coalitions - initiated by potential global standard setters with the explicit purpose of locking-in as many firms as possible into their proprietary product, architectural, or interface standards. This is the case of Wintel (Microsoft and Intel association explained on the chapter 2.2.4) and .Net (discussed on chapter 3.2.2).

- Technology cooperation networks – built to facilitate the exchange and joint development of product design and production technology, involving cross licensing and patent swapping, and permit the sharing of R&D. Under such arrangements, knowledge typically flows in both directions and all participants need to master a broad array of technological capabilities. The mainframe architecture (explained on the chapters 2.1.1 and 2.2.1), MDA (discussed on chapter 3.1.1) and Java (discussed on chapter 3.2.1) illustrate this.

5.3. Peer-to-peer and collective conscience

"Some people use virtual communities as a form of psychotherapy. Others spend many hours per day pretending they are someone else, living a life that does not exist outside a computer."
- Manuel Castells

In parallel with the formation of the network enterprises, academic people started to use the network to build strong communities of interest, which started by exchanging ideas and finally discovered a potential to defy and compete big monopolies. New companies were created to exploit the new technologic developments in ways unanticipated by the scientists and big companies - it was the case of Apple, Microsoft and Intel – reducing the entry price to be part of the network. The benefits

[238] Source: Bibliography 1
[239] *in* Bibliography 1, pages 176, 178 and 210.
[240] See bibliography 60, chapter 4.3

of being in the network grew exponentially, because of the greater number of connections. This gave birth to the modern counterculture movements like the hackers and the crackers; it helped to break geographic barriers in the consolidation of currents of thoughts, like anti-globalisation; it helped the formation of worldwide/underworld organizations like the (cyber) terrorism.

5.3.1. Online communities

One of the start points of the online subculture was the set of science networks (like ARPANET, CSNET, BITNET) being used to exchange personal messages around subjects like science fiction, but the advent of personal computing and cheap networking equipment gave birth to the BBS (Bulletin Board Systems). Described by Rheingold as "a grassroots element to the Net that was not, until very recently, involved with all the high-tech, top-secret doings that led to ARPANET (…). Real grassroots, the kind that grow in the ground, are a self-similar branching structure, a network of networks. Each grass seed grows a branching set of roots, and then many more smaller roots grow off those; the roots of each grass plant interconnect physically with the roots of adjacent plants, as any gardener who has tried to uproot a lawn has learned."[241]. Rheingold – who was an active member from an online community called WELL[242] and specializes today in the creation of new communities[243] - uses the term virtual communities to define the social phenomenon spawned from the BBS. Virtual communities are social aggregations that emerge from the Net when enough people carry on those public discussions long enough, with sufficient human feeling, to form webs of personal relationships in cyberspace.

He identified that the technology that makes virtual communities possible has the potential to bring enormous power - intellectual, social, commercial and political - to the citizens, but this latent technical power must be used intelligently and deliberately by an informed population. He considered the online subculture to be "like an ecosystem of subcultures, some frivolous, others serious."[244] Some of the serious communities united programmers, who used the network to exchange information and programs – mostly related to the network itself – aiming recognition from their peers.

The hackers were already used to the time-sharing systems networks as a communication medium. With the networks, it was easier to exchange programs and routines, and the university walls were not barriers as before. The community would then span to other academic centres, and to other countries. Due to fate, or to the anarchic tendencies of academic ecosystems, the networks started to be organized in a horizontal manner.

[241] in bibliography 20
[242] http://www.well.com/. Rheingold also connected to other communities like TWICS, CalvaCom and Usenet (http://www2.webmagic.com/usenet.org/)
[243] See http://www.highermind.org/rheingold/Associates/
[244] in bibliography 20

5.3.2. Peer networks and cooperative computing

Initially, the term peer-to-peer (P2P, or simply peer[245]) described a protocol, application, or network where every node had equivalent capabilities and privileges, being able to initiate or complete any supported transaction. Beyond the technical definition, the term started to designate decentralized virtual communities where every individual participated in the same level, obtaining information with the same access rights, and sharing this and new material with other network members. Bauwens[246] abstracts the peer-to-peer concept to other levels, like politics and spirituality, and even suggests the hypothesis of a new civilization format based in P2P[247].

Bar and Borrus suggested that based on the two elements already discussed - ubiquitous computing and a coherent infrastructure - a new computing paradigm emerged in the 1990s, shifting from simple linkage of computers to "cooperative" computing[248]. Mimicking the cooperation between different companies, and based on peer-to-peer anarchic structure of organization, the hackers started to organize themselves, initially simply letting spontaneous and informal communication flourish at the same time, generating reciprocity and support by the dynamics of sustained interaction.

5.3.3. May the force be with the hackers

The world-wide web was built on the contribution of the hacker's culture of the 1970s. The group of researchers at CERN led by Tim Berners-Lee and Robert Cailliau relied on the hypertext concept created by Ted Nelson in the 1970s[249], used hacker technology like UNIX and TCP/IP and distributed their software free over the Internet. Richard Stallman and Linus Torvalds gave the initial impulse to the new-hackerism movement, by creating GNU and Linux. As important as GNU and Linux technological features, are their sociological ones. They are in the core of the modern *hackerdom* activities. They started the projects alone, and once their concept was stable enough to be understood by other people, they used the Internet as the communication media to form P2P task forces, by opposition to having a project team - sponsored by governments or companies – working in the same space. They had a centralization role, needed to guarantee the integration among the different software parts, but the community was still in the power.

According to Eric Raymond, "Linux is subversive. (…) I believed that the most important software (…) needed to be built like cathedrals, carefully crafted by

[245] A good definition may be found in http://www.wikipedia.org/wiki/Peer_to_peer
[246] http://users.skynet.be/michel.bauwens/index-2.html
[247] For more information, read the interesting paper in bibliography 31
[248] See the interesting paper in bibliography 62.
[249] In Dream Machines, Nelson provides three categories of hypertext: The first, basic or chunk hypertext, supports what we have been calling reference and note links. The second, stretchtext, is a full implementation of expansion links. The third, collateral, stems from his work in 1971 with the Parallel Textface, which provides a view of two documents on one screen, with full support for versioning. Nelson also distinguishes between "fresh" or original hyperbooks on one topic, "anthological" hyperbooks linking different works, and "grand" systems. Source: bibliography 63.

individual wizards or small bands of mages working in splendid isolation. (…) Linus Torvalds' style of development — release early and often, delegate everything you can, be open to the point of promiscuity — came as a surprise. No quiet, reverent cathedral-building here — rather, the Linux community seemed to resemble a great babbling bazaar of differing agendas and approaches (aptly symbolized by the Linux archive sites, who'd take submissions from anyone) out of which a coherent and stable system could seemingly emerge only by a succession of miracles."[250]

5.3.4. The motivated and ethical hacker

The most difficult, for people outside the hacker communities, is to understand what are their motivations. According to Himanen[251], the hacker ethic may be divided into the work ethic, the money ethic and the network ethic. By the work ethic, he explains that the hacker activity must be joyful, enthusiastic and passionate, while performed in an individualistic rhythm of life.

The money ethic states that the main hacker objective shall be the recognition from the peers, and through the "capitalism hackerism", one can take part in the traditional capitalism only temporarily (until have enough capital to dedicate exclusively to "have pleasure") or on a part-time basis (working for a traditional company during the day, developing free software during the night). In addition, most of the hackers who established companies to earn money from free software and open source don't see any problem in selling software or services, once the work ethic is followed.

The network ethic (or nethic) preaches that a hacker should always try to practice the freedom of expression, respect privacy and stimulate self-activity – "the realization of a person's passion instead of encouraging a person to be just a passive receiver in life (…) very different to the traditional media"[252]

"A hacker who lives according to the hacker ethic on all three of these levels gains the community's highest respect. This hacker becomes a true hero when she or he manages to honour the final value (…): creativity - that is, the imaginative use of one's own abilities, the surprising continuous surpassing of oneself, and the giving to the world of a genuinely valuable new contribution"[253].

5.3.5. May the force be with the hackers

David Stutz, The man formerly responsible for Microsoft's anti-open source strategy, attacked in February 2002 (just after his retirement) Microsoft's PC-based strategy, which he argued as misguided in a computing world where complex networks are more important than single devices. He maintained that the internet, the web and open source software projects, in which communities of programmers contribute improvements which are distributed free, are all part of the steady advance of networked computing.

[250] Raymond's chef d'oeuvre, The Cathedral and the Bazaar is available online. See bibliography 6.
[251] See bibliography 2
[252] See bibliography 2, page 106.
[253] See bibliography 2, page 141

Stutz suggested that Microsoft needs to focus on building a layer of software that integrates network technology. But that layer should not be an operating system like Windows, which is tied to PC technology. "To continue to lead the pack, Microsoft must innovate quickly," he said. "If the PC is all that the future holds, then growth prospects are bleak."[254]

5.3.6. Lingua Franca

The hacker communication tools are the e-mail, newsgroups, chats and, later, the weblogs. The contents are around new programs, tools, routines, problems, and updates. The language was often English. It changed, after the spread of cheap web access in other countries, and then voluntary translations for programs, web sites and documentation flourished – again in exchange of recognition. This phenomenon helped to expand the hackerdom borders, by creating hacking sub-networks speaking regional languages, even if the common language among different local communities was English. To some analysts, the exchange of information via written medium could give an impulse to the recuperation of the constructed and rational discourse. What finally happened, on the contrary, was the stimulation of a new form of language, expressed by the electronic texts and richly completed by funny symbols, weird acronyms and multimedia.

Another important factor is the asynchronous communication favoured by the e-mails and newsgroups. This allowed each developer to work when convenient, with no time obligations, exactly as preached by the hackers ethic.

5.4. Privacy

The need for privacy on the net started to attract the public opinion in 1999 during the menace, from Intel, to create a processor identified by a unique number, known as PSN (Processor Serial Number). Although this practice is current in mainframes – software licenses are often validated by comparing an encrypted key with the computer's serial number – its implementation in the personal computers could be the foundation of a vast tracking system that could help accumulate data on users as they travel around the Web, violating their fundamental right to privacy. The outraged privacy advocates launched a boycott of products containing the Intel Pentium III chip, the first such broad-based boycott of a product over the privacy issue[255]. And they won the battle. After a letter from the American government[256], Intel finally decided to disable the PSN feature[257].

Another long-term battle is against the abusive usage of cookies[258], to trace the virtual footsteps of online users. The main advantage of cookies is the addition of a

[254] Source: Bibliography 82
[255] See http://www.bigbrotherinside.org/
[256] Representative Edward Markey (D-Massachusetts), the ranking minority member of the House Telecommunications, Trade, and Consumer Protection subcommittee.
[257] See bibliography 66
[258] According to Netscape, cookies are a general mechanism that servers can use to both store and retrieve information on the client side of the connection.

simple, persistent, client-side state, which significantly extends the capabilities of Web-based client/server applications. Without this type of persistent applications, it is virtually impossible to securely transfer the user information between two web pages, functionality required by commercial web sites. Nevertheless, sometimes the collection of information is excessive, with consumer habits being monitored by marketing companies and stored into databases, later used in aggressive advertising actions.

The original cookie definition, by Netscape, had several flaws, avoiding the user acceptance of the cookie execution. The IETF prepared a new proposal, containing a privacy section to enforce the need for this acceptance[259]. The newest releases from Microsoft explorer implement part of those suggestions.

A famous case is the EPIC against DoubleClick. “EPIC (Electronic Privacy Information Center)[260] filed a complaint[261] with the Federal Trade Commission on February 10, 2000, concerning the information collection practices of DoubleClick Inc., a leading Internet advertising firm, and its business partners. The complaint alleges that DoubleClick is unlawfully tracking the online activities of Internet users (through the placement of cookies) and combining surfing records with detailed personal profiles contained in a national marketing database. EPIC's complaint follows the merger of DoubleClick and Abacus Direct, the country's largest catalog database firm. DoubleClick has announced its intention to combine anonymous Internet profiles in the DoubleClick database with the personal information contained in the Abacus database.”[262]

Important is to notice that all discussions about cookies, and the discovery of the privacy issues, are due to the openness of the HTTP protocol, needed for the interoperability between the HTML language and the browsers. If similar initiatives were taken under proprietary environments, everything could remain secretly hidden for a long time.

[259] See bibliography 68.
[260] http://www.epic.org/
[261] Available on http://www.epic.org/privacy/internet/ftc/DCLK_complaint.pdf
[262] in http://www.epic.org/privacy/internet/cookies/

6. The New Economy

"The concept of profit has always been the noble version of a deeper, more fundamental human instinct: greed" - Manuel Castells

A new economy – networked, global and informational – emerged in the last quarter of the twentieth century on a worldwide scale. Its epicentre has been the information technology industries and financial institutions in the 1990s. The first aspect of this new economy – the network – has been discussed in the previous section. The second aspect is the institutional, organizational, and technological capacity to work as a unit in real time, or in chosen time, on a planetary scale, enabling the company to survive and thrive in this global economy. This discussion is out of the scope of this document. Let's now consider the fundamental elements of the third aspect – the informational economy – always from the open source and open standards point of view.[263]

The quest for possible answers to three important questions will drive this chapter:

- What strategies have been used by the IT companies to dominate the hardware and software market?
- What is the impact from the open platforms and standards on those strategies?
- What are the chances for the Open Source community to increase the market share?

To increase profits, when looking for short-term results, there are four main ways: to reduce production costs (as seen in the last two years with the labour cost reduction - mainly in the ICT market – and the falling price of electronic components), to accelerate capital turnover, to broaden the market, and to increase productivity. This last has been the main advantage of the open source products over the competition. The productivity, when applying the hacker ethical principles[264], is extremely higher than in the other firms[265]. Its technology is proved to have higher quality and to be more innovative. Productivity and technological innovation are important means, but certainly not the only ones.

6.1. Standard wars

When a new technology standard appears, it may complement (and sometimes merge) or compete with the existing ones. When UNIX become a standard for open operating systems and started to dominate the market, IBM decided to create its own

[263] The globalisation of the financial markets is considered the backbone of the global economy, and together with the speed of market operations, it is possibly the main cause of the recession started in 2001.
[264] Source: Bibliography 4
[265] Source: Bibliography 1, page 95.

version of UNIX, and let the mainframe platforms to be UNIX compliant. It reacted so fast that the OpenEdition layer of the MVS operating system was one of the first to comply with the POSIX standard. This was motivated by two interdependent factors: The first was the risk of the established base of mainframe customers to migrate from the old systems to the most attractive and flexible UNIX environments, which appeared to be less expensive and to have more human knowledge resources coming fresh from the universities. The second was the usage of UNIX-compliant software under mainframe platforms. It revealed to be less expensive for IBM to develop a UNIX layer than to redevelop a whole subset of software – mostly related to the Internet, Web, User interfaces and Database technologies - to compete with the UNIX versions. It proved to be the right decision – even if many customers eventually migrated to open systems platforms – with many mainframe customers using today the UNIX layer to run network and Internet services.

With Microsoft, it was the opposite. At first, in the desktop segment, when its cooperation with IBM and Apple was terminated, it decided to profit of its own installed MS-DOS base – a high-valued network – to implement a new graphic environment. The quality revealed to be a less important element than the number of users (see section *Network externalities below*), and quickly Microsoft became the preferred operating system provider from the manufacturers of PC-compatible machines – including IBM. In the server segment, Microsoft always thought of mainframes and UNIX as legacy technologies, and started by dominating the low-entry market share, by providing small servers fully compatible with the installed base of desktop PCs. With the increased capacity of Intel-based machines, the Microsoft servers started to perform functions for which a mainframe was needed before. And Linux appeared to play in the same market share than Microsoft windows servers. As the ideals of Microsoft and the Open Source community were completely different (money, monopoly and marketing against gratuity, freedom and quality) the fusion or cooperation was not possible.

6.2. Network externalities

As Metcalfe helped us to demonstrate in the section 5.1, the value of the network increases with the number of its nodes. All new networks start with a zero value, and the network externalities from the opponent networks make their spread more difficult. When a new technology appears, incompatible with the existing ones dominating the market, a huge effort must be made to create a large network, for its value to be higher than the changing costs[266]. This concept is not new, and has been implemented in the postal services, railways, airlines and telephones.[267]

Currently, in the operating systems market for desktops, the value of the network composed by the windows users is extremely higher than the competition, as more than 90% of the users are connected to it. This high value forces most of the companies and individuals willing to use Macintosh or Linux desktops to keep at least one PC compatible with the windows systems. It also motivates the creation of Linux products that may interface with windows systems – by recognizing the Microsoft proprietary formats, by being able to execute windows-compatible programs in a simulated environment, by recognizing the Microsoft proprietary protocols in client-server and network connections. This causes negative reactions from Microsoft,

[266] See section 6.4, paragraph *Changing costs* on page 104.
[267] See bibliography 7, chapter 8 for a more complete discussion. See also the bibliography 1.

which keeps changing its standards and protocols to difficult the connections and data exchange with other systems and products, aiming to keep the users locked into its network, and forcing them to pay expensive costs to get out.

When a company has a complete monopoly in the operating systems or hardware market share, it is extremely easy to trigger the network externalities in favour of its own related products to destroy the competition. It was always the IBM strategy in the mainframe market. A first example is the security layer, tightly related to the operating system to improve the enforcement of the access authorization. IBM has its own product – RACF, later renamed to Security Server – that comes imbedded in the MVS installation material, even if not ordered. To install the products from IBM competitors, one must follow a procedure to remove the IBM product. A second example was in mainframe networking. A company called Interlink developed a software product that implemented TCP/IP on IBM mainframes. The performance was better, the price lower, and the flexibility higher, the door to the Internet was finally open. IBM quickly offered its own TCP/IP stack “free” together with the operating system, and started to build tight links between its own product and its databases. After complaints from customers, IBM started to provide a refund for companies that did not want IBM TCP/IP. As the impact was only felt by big companies, the affair is almost forgotten. A similar case happened with Microsoft against Netscape in the browser segment. This time, with a larger publicity and the difference that Microsoft continues to freely distribute Internet Explorer, what practically eliminated Netscape from the browser market[268]. The only solution found by Netscape was to “escape” to the open source community, creating Mozilla, one of the best browsers compatible with Linux[269].

6.3. Feedback

The feedback is an important factor in the competition for market shares, already existent in the industrial age, which attained a fundamental position in the information economy. Due to the virtualisation of the networks explained in the section 5.1, the feedback effect tends to create monopolies in each domain of the software and information realms. The feedback may be classified, according to its impact for the creation of monopolies, as positive or negative. Please note that the feedback impact for the economy as a whole might be the opposite than its name would suggest.

[268] Description of this case may be found on bibliography 18 (page 99).
[269] Full story may be found on the bibliography 5 (page 197).

6.3.1. Positive

The feedback is tightly related to the network externalities. In cases of positive feedback, the quality of a product (or sometimes the success of a brand or its marketing strategies) expands the network of users, which will in turn increase its network value, motivating more users to integrate it. This may happen successively, in a positive spiral, until the stability brought by the saturation level of the market segment. The earlier the product starts its positive feedback, the more chances it has to become a monopoly. However, as the technology evolves in cycles, products dominating the market in one moment can be quickly dominated in the next evolution step if they are unable to evolve as quickly as its opponents, or if they cannot react to new products bringing superior features.

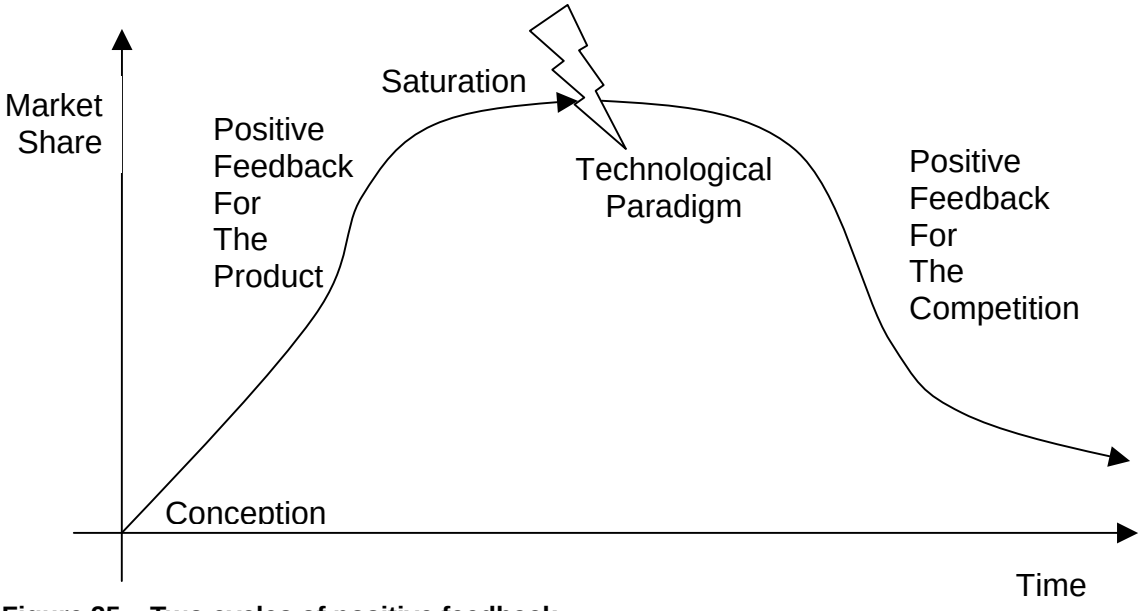


Figure 25 – Two cycles of positive feedback

To illustrate this, let us consider the Figure 1 above and the computers market from the 1960s until the 1990s, discussed in the section 2.2.1. Initially there were several computer suppliers, competitively sharing the market. Then IBM created the System/360 architecture, with a superior technology, standards and aggressive marketing campaigns. It quickly dominated the market, eliminating part of the competition, and forcing the remaining to survive with a small market share. The paradigm was the evolution of UNIX systems, which could provide a good an inexpensive solution, as demanded by the potential customers. The market share of the IBM machines - then known as mainframes - started to reduce, forcing IBM to evolve an old platform – S/36, then transformed into AS/400 –, create its own UNIX-compliant machines and systems – AIX – and create a UNIX compatibility layer in the mainframe systems. The mainframe environments are targeted for death since then, but elevated migration costs and unbeatable technical superiority are still keeping them alive with a stable market share. IBM is trying to stimulate a new positive feedback in favour of their mainframes – now called high-end servers – with a strategy called server consolidation, which includes a compatibility with the Linux operating system and the advantage of integrating the processing capacity of multiple low entry servers into a single box.

6.3.2. Negative

Shapiro and Varian[270] may describe this later reaction from IBM as a negative feedback, when force motivates weakness (UNIX competition conquer market share from IBM mainframes) and weakness motivates force (IBM redesign the mainframe systems to be compatible with UNIX and Linux). If IBM succeeds in its movement, the result may be the stabilization of the market, until the next paradigm, as shown in the Figure 26 below:

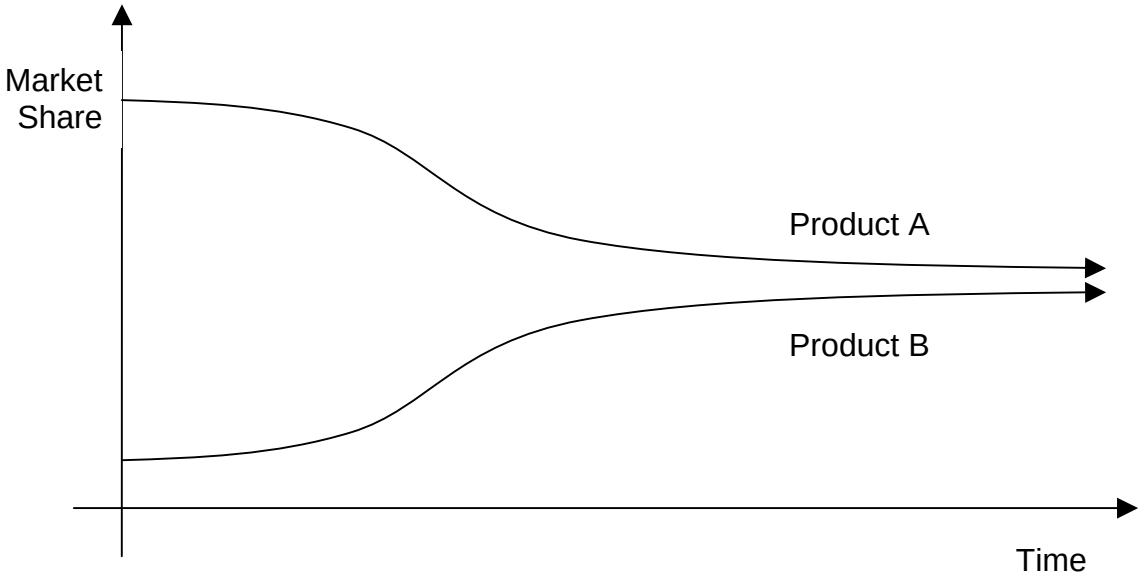


Figure 26 – Negative feedback

Standards de facto often evolve in the marketplace favoured by positive feedback cycles. As seen, they stimulate the monopolies in detriment to a healthy competition. The absence of competition normally has negative impacts in the technology (by reducing the pace of innovation), the social welfare (by increasing unemployment rates with the elimination of the competitors) and the economy (by directing the flows of profit into the dominating company).

6.3.3. The role of the standards

Standards de jure can promote the competition, by creating a negative feedback with the publication of the standard protocols and definitions, the implementation of several compatible products, and stimulating the innovation by the different implementations of the standard recommendations. This is one of the reasons by which the standards are increasingly limited to general specifications, leaving the details for the companies and scientific community to decide. The other reason is to reduce the time needed to develop a standard, allowing the negative feedback effect to happen before one company decides to implement its own product, outside the standard specifications, trying to quickly ignite a positive feedback in its favour. This

[270] See bibliography 7, page 158.

is the preferred practice of companies like Microsoft, as defined by Bill Gates in his book *The road ahead*: "Because de facto standards are supported by the marketplace rather than by law, they are chosen for the right reasons" [271].

Microsoft is also accused of practicing "vapourware"[272]: to avoid the competitors to conquer its market with products offering better quality, creating a technological paradigm and originating a positive feedback, one company may announce a new version of its product, containing the same facilities than the competition, even before its design.

6.4. Cost Analysis

6.4.1. Production Costs

The most important part of the production cost for software development, is from research and development (R&D). Other important costs are for marketing, promotion and documentation.

Those costs are important even for the open source software, despite the wrong impression that they are "free". Studies based in the COCOMO model[273] estimates that "It would cost over $1 billion (…) to develop [the GNU] Linux distribution by conventional proprietary means in the U.S."[274]. Another study points out: "If Debian [one of the Linux distributions] had been developed using traditional proprietary methods, the COCOMO model estimates that its cost would be close to $1.9 billion USD to develop Debian 2.2"[275]. IBM announced investments of $1 billion USD in R&D for Linux, many other hardware and software suppliers – including HP, Intel, Sun and SAP participate in Linux research laboratories.

6.4.2. Reproduction Costs

As for the other products from the digital information economy, the reproduction costs are extremely low. They are normally the duplication of a CD-ROM, and may imply the reproduction of documentation.

The reproduction costs of the open source software, if we consider the main Linux distributions, are even higher than the proprietary software. Normally the Linux kernel is supplied with many complementary software products, in several CDs, and a good documentation for the installation and customisation procedures.

[271] See bibliography 135, chapter 3.
[272] See bibliography 135, page 98.
[273] See the description of the method in the bibliography 73
[274] For the complete study please refer to the bibliography 73
[275] For the complete study please refer to the bibliography 74

6.4.3. Distribution Costs

The open source software may be distributed in two ways: a free download from the web (with indirect network costs for the distributors and users) or a CD containing the operating system, tools, applications and an installation manual. In this second way, it has the same costs than other proprietary products.

6.4.4. Transaction Costs

The transaction costs are measured by the cost for the customer and the distributor per copy of the product.

Probably the main difference between Linux and the other operating systems, and between open source software distributed under the GPL license and other commercial software is the reduction of the transaction costs, for companies installing them in several computers. Normally commercial software is licensed according to the number of computers or users. The open source software is often distributed without limits for its usage, copy or installation. This is extremely important for big companies, and in moments of financial crisis, this might be a determinant factor for the adoption of open source software. In parallel, the new licensing practiced by Microsoft is helping to push customers out of the windows software, mainly from its server family. This has been anticipated by Bill Gates in 1995: “Customers express to me their worry that Microsoft, because it is, by definition, the only source for Microsoft operating-system software, could raise prices and slow down or even stop its innovation. Even if we did, we wouldn’t be able to sell our new versions. Existing users would not upgrade and we wouldn’t get any new users. Our revenue would fall and many more companies would compete to take our place.”[276] This may reveal an even more amazing approach: Microsoft may have increased its licenses to measure the real interest by the companies in the open source software. If it starts loosing market shares, its prices may be reduced again, to the previous level, in parallel with a strong marketing campaign.

6.4.5. Changing costs

The changing costs are paid by the companies and individuals willing to adopt a new technology, changing from an existing network of users to another. The total changing costs are calculated by adding the investment in new hardware, software, training and the maintenance of the compatibility with suppliers and customers. For example, companies wishing to move from windows to Linux benefit from the first two elements (Linux is compatible with all windows hardware with better performance, and the license and maintenance fees from open source products are usually lower). They may have expenses with training, higher for the people responsible for the installation and support than for the end-users. These subjects will be detailed on the next paragraph (*TCO*).

[276] in bibliography 17, page 63.

When analysing the implementation of a new technology inside a company or intranet application, only the internal costs are considered. When the implementation implies changes in external users, the collective changing costs must be taken into account. An example can be taken from the traditional client-server applications, when part of the application runs in the client side, another part on the server. If a new release of the server component - requiring a new version of the operating system - is to be installed, the changing costs are limited by the reduced number of servers. If a new release of the client component needs a new version of the desktop operating system, the cost is multiplied by the number of users. In case of applications available outside the company – like clients for home banking applications - the situation is even more complicated, because the customers need to be convinced to upgrade their own operating systems. Internet applications helped to eliminate this problem, with the usage of server applications exchanging information with the clients by the usage of standard formats and protocols – like TCP/IP and HTML. If a new version of the application is released, only the application servers are impacted. The customer is even free to choose the more convenient operating system.

This is also good a reason for the organisations controlling the Internet names (like the URL) to remain non-profit. The collective changing costs of changing the URL of a web site will imply the usage of publicity, electronic and paper communications, change of cross-links with other web pages, and may often imply the loss of customers coming from old links. If a single – and commercial – institution keeps the control of the URLs, the risk of abusive prices and practices is extremely high.
High changing costs are normally considered as a lock-in situation. Shapiro and Varian identified seven types of lock-in situations, frequent in the information economy[277]: Contract obligations, Durable goods, Specific training, Information and data bases, Specialized suppliers, Research costs, Fidelity programs. Important for our study are:

- Durable goods – The usage of open platforms may reduce the risk of the technological lock-ins (if the equipment is changed, the software and applications must follow). The standards are often implemented by several hardware and software companies, allowing the replacement of determined products with lower changing costs than proprietary alternatives. Commercial lock-ins (when the compatibility only exists between complementary products from the same supplier or from a close network of “authorized vendors”) may be eliminated with the usage of open standards. The implementation of open source solution may also eliminate the need for the implementation of new hardware, as discussed below.

- Information and data bases – The usage of strictly open formats – like HTML and XML – may ensure the compatibility with many software components, and eliminate the need for conversion tools when the application is replaced. When proprietary databases are implemented – like Oracle and DB2 – the existence of tools to convert the information into open formats must be verified, and the work needed for the data extraction, conversion and insertion into the new database might be taken into account.

- Specialized suppliers – Open solutions are normally supported by many different companies, which have free access to the standards, protocols, documentation

[277] Translated and adapted from bibliography 7, page 109.

and sometimes even the source code. This is essential to ensure good problem solving techniques and a total independence from the services provided by the supplier.

The lock-in situation has also an impact in the suppliers, which are forced to adapt to new strategies from preferred customers or partners. This has been the case recently for software companies developing applications using DOS interfaces, when Microsoft decided to disable a subset of DOS functionalities in the Windows XP family.

Unification strategies like the single UNIX specification discussed on chapter 2.2.2, and the ongoing United Linux initiative (chapter 2.2.3) aim to reduce the lock-ins provoked by different (and incompatible) implementations, and to minimize the impact for the application suppliers when new versions of the operating systems are released. They also reduce the cost for application and hardware suppliers, which have a single version to test and validate their products.

The implementation of the Internet, based on open standards and protocols, is considered to be a turnkey for its development, with the consequent perfect communication among different hardware and software suppliers, and the absence of lock-in situations.

6.4.6. TCO

A difficult task when comparing alternatives – in our case proprietary against open source – is the need to estimate all the costs involved in the implementation (changing costs) and the recurring periodic costs. This is called "Total Cost of Ownership"[278]. Elements like operating system licenses and hardware equipment are obviously included, but salary, consultancy services, training and the licenses for complementary – although mandatory – software should be obtained in advance. The scope of this document does not allow an extensive study of all costs implied in the installation and maintenance of servers and desktops; several comparisons have already been made[279] and a brief analysis may be summarized by some of the TCO items:

- Hardware – Solutions built around environments like Linux and Windows can use the same hardware equipment, and even if the performance obtained by Linux is commonly accepted to be better, this can be ignored for a fair TCO comparison[280]. The comparison with proprietary hardware solutions, like Sun, Apple, IBM mainframes and AS400 generally gives a higher difference, in favour

[278] TCO (total cost of ownership) is a type of calculation designed to help consumers and enterprise managers assess both direct and indirect costs and benefits related to the purchase of any IT component. TCO analysis originated with the Gartner Group several years ago and has since been developed in a number of different methodologies and software tools. Source: search390.techtarget.com

[279] Good comparisons may be found in bibliographies 75, 76 and 77. Another good collection on articles on Linux vs. Windows TCO may be found on http://searchwin2000.techtarget.com/featuredTopic/0,290042,sid1_gci845125,00.html

[280] NB – The argument here is: what we cannot clearly measure, we shall not use as a comparison factor.

of the open source alternatives[281]. Hardware maintenance fees, insurance and floor space may also be considered to obtain a more complete analysis.

- Software – The Open Source solutions are always cheaper than proprietary ones, for small-to-medium environments. Generally, the license costs for large environments should consider the usage of proprietary databases and applications[282], which are generally less expensive – or with an equivalent price - when installed under open source environments. An important factor to consider is the need to pay yearly fees – known as maintenance and support – to have the right to use the software.

- Human resources – It is extremely difficult to estimate the manageability – or the simplicity of installation, configuration, support and usage - for each solution. Commonly accepted arguments for solutions based on Windows are the easy installation (it is normally pre-built into new hardware) and cloning – duplication of servers and desktops -, absence of structured problem-solving analysis, and consequently the need for more support and help-desk people, and operations like “system restarts” and recovery of lost information. For Linux and other UNIX systems, the existence of well-trained (and more expensive) people, the time-consuming tasks to install and configure the environment, with later facility to maintain it. IBM mainframe environments are far more difficult to install and configure, often requiring specialized consulting services, but easier to maintain after the stability level is attained. Smaller systems tend to require more servers to give the same capacity of large systems, and consequently need more people to support them[283].

- Training – The clear advantage is for well-known products like Windows, which are still largely used in schools and at home. The Linux interfaces – like KDE[284] - and office suites – like OpenOffice[285] and StarOffice[286] - are quickly reducing this advantage[287]. For the support staff, the time needed to learn how to maintain an open or proprietary environment is not different. Exception made for full proprietary environments, like IBM mainframes and AS400, where the complexity is higher, and so is the time for the technical staff to be fully productive. For any solution, the ideal situation is often to have experimented consultants to help in the initial steps and to complement self-study and classes with ad-hoc training.

- Services – Related to the training costs, the consultancy services needed to help in the implementation of any solution are more dependent on the complexity of the environment than the openness of the source or platform. There are always differences in the price for consultancy for each platform, but due to the volatility

[281] NB – The decision upon the implementation of a full proprietary solution (hardware tied to the software) normally consider other factors than cost.
[282] Some examples: DB2, Oracle, SAP, SAS.
[283] NB – New technologies that help to establish a single and central point of control for several servers, and to easily clone environments are changing this scenario.
[284] http://www.kde.org/
[285] http://www.openoffice.org/
[286] http://wwws.sun.com/software/star/staroffice/6.0/
[287] NB - Even if open source suites are able to read the simple documents, some features – like imaging, indexation, forms - can be nastily converted. This is normally attributed to the lack of portability of Microsoft formats, which also make impossible to guarantee their perfect visualization in the different printer and screens. However, it was always the case of Microsoft products, and this is not due to change.

and dependence of cultural differences, this should be analysed in the moment and country of the implementation.

6.4.7. Cost comparison

The overall conclusion - based on the previous paragraphs and on the bibliography referred by them - is composed by the following points:

- For each item described above, the initial costs, the changing costs and the recurring costs should be identified.

- Considering the installation of new hardware and software, in a company or educational establishment without previous knowledge of any solution – open or proprietary – the TCO for implementing proprietary software is more elevated than any open source solution, with the difference being directly proportional to the size of the computing environment.

- As most of the companies already have an installed base of computers, the changing costs might be more important the other elements, and must be analysed carefully. Conversions from proprietary environments like mainframes may have a large cost for the redeployment of the applications and the tasks needed to redevelop the interfaces with other platforms. Migrations from Windows-based environments to open systems shall have a higher transition cost for the desktops than for the servers – due to the reduced impact on end users -, so the size of the network is an important factor to consider.

- To compensate the changing costs, the recurring costs after the initial investments are generally lower for open environments, thus when considering the cost impact for a period like 3 to 5 years after the implementation of the new system can give different results than comparisons taking into account only the implementation costs.

- All environments must be continually upgraded to keep current operating system versions and to profit from technological evolutions, like improved security, increased processing capacity, reduced machine size. The cost for upgrading mainframe environments is known to be high in terms of human resources – but decreasing – and normally oblige hardware changes each 10 years[288]. Windows-based systems, while having usually minor technological advances, imply hardware changes – or upgrades of memory and processor resources – each 2,5 years. In the open systems, the capacity is directly related to the amount of service needed and not to the version of the operating system. An average time between hardware upgrades is estimated to 5 years, to benefit from the technological advances.

- When the changing costs are higher than the cost savings, and the technological benefits are not compensated by attaining important business objectives, there is a lock-in situation. The decision is complicated because of the recurrence of the consequences: The lock-in increases with the time, due to the increasing usage of the technology – by new people, in new functions and by new applications -

[288] See paragraph Architecture on page 21.

and consequent higher changing costs. This is one of the best arguments in favour of the open standards: freedom of choice. Once the migration is done, the changing costs may be amortized in the long term, and the company is free from lock-ins. If the supplier increases the price, does not follow the evolution pace or does not give a good support, the changing costs for a new alternative are lower: the open formats and standards are compatible with several products from the competition and consequently the migration tasks and training become easier. This is an example of good recursion: as the supplier is aware of the facility to change, it is more motivated to ask for a correct price and continuously invest in innovation. Both supplier and customer benefit from this process.

	New Installation	Changing Costs	Recurring Costs	Upgrade Costs	Lock-in risks	Freedom of choice
Open	↘	↗	↘	↘	↘	↗
Proprietary	↗	↘	↗	↗	↗	↘

Table 3 - Cost comparison summary

6.4.8. ROI - The conclusion is beyond the costs

Despite the obvious reduction of software costs and considering the high changing costs and the nightmare of calculating a real TCO, the Open Source community tries to concentrate their marketing efforts in comparing open source against proprietary software (and the example is often Windows x Linux) by measuring factors like manageability, control, stability, scalability and freedom of choice. This may be represented in a business case by estimating the amount for the Return on Investment[289]. Besides the cost savings obtained, which in our hypothesis is already part of the TCO, all the other elements benefit the open source software, and the analysis should concentrate in the importance of the benefit for the business objectives.

As an example of the indirect benefits, let us consider the opinion from Bruce Perens: “Control means being able to get a different service provider if you don't like the service you're getting on your software. Control means not having to convince the software's producer that your needs fit in their marketing plan. Control means not living in fear that the BSA (Business Software Alliance) will bring federal marshals to raid your business. Control means not having a domineering software company”[290]. More benefits can be exploited with the help from the explanations and comparisons made on chapter 2.

A careful functional analysis should always precede the cost analysis, to verify if the open source alternatives can be implemented. A good example is for graphic environments. Currently there are no good solutions to compete with proprietary software – which largely uses proprietary formats - like Macromedia Flash®[291], 3D

[289] For a given use of money in an enterprise, the ROI (return on investment) is how much "return," usually profit, cost saving, or indirect benefits that help to satisfy business needs. See http://www.rms.net/lc_faq_other_roi.htm

[290] in bibliography 72.

[291] http://www.macromedia.com/software/flash/

Studio Max™[292] and Adobe Photoshop®[293], which only run under proprietary platforms (Mac or Windows).

Despite the high importance given by consulting reports to the ROI evaluation, a research conducted by DataNews showed that only 30% of the Belgian companies measure the ROI to verify the economical benefits of technological projects.[294]

[292] http://www.discreet.com/products/3dsmax/
[293] http://www.adobe.com/products/photoshop/
[294] Source : DataNews n°02, 17/01/2003, Page 4

6.4.9. Case studies – cost reduction

The Linux strategy is a major part of Unilever's drive to cut its IT bill, a part of an organisation-wide plan that already reduced the IT budget from €600m (£398m) in 2000 to €500m (£332m) in 2003, and aims further reduction of €100m until 2006. The main savings will come from hardware, which currently accounts for 40 per cent of the company's infrastructure costs. The Linux strategy will contribute through server consolidation and by reducing unit costs.[295]

Implementation of Linux is also behind cost saving for Morgan Stanley. The numbers have not been announced but the company said that the goal of the project, underway since mid-2001, is to reduce the cost of computing by adding flexibility to its computing architecture. Morgan Stanley's institutional securities division has opted for new architecture that moves the company's data, applications and operating systems off specific machines and onto network servers throughout its global computing infrastructure.

This infrastructure is made up of around 6,000 servers, 25,000 desktop computers and thousands of applications. The aim is also to increase flexibility : "We want to be able to run any application on any box at any time," said Jeffrey Birnbaum, managing director and global head of enterprise computing at Morgan Stanley's institutional securities division. "Now, around 35 per cent of our servers are running Linux."

By 2005, two years ahead of schedule, the division plans to be running 80 per cent of its systems on commodity hardware, most of them using Open Source and open protocols for hardware independence. "When you run everything including the operating systems off the network adding computing power is simple," explained Birnbaum.

A study published by the Swiss consultancy Soreon Research GmbH concluded that “Companies with a €1 million ($1.1 million) budget for office software can reduce their costs as much as 20% by using OpenOffice software instead of Microsoft's Office product. (…). And those running the open source Linux operating system instead of Microsoft's Windows on their servers can save as much as 30%”. An interesting conclusion of this study is that small and midsize enterprises may benefit only marginally by using open source software. “A company with 10 computers, for instance, can reduce its costs around 2% by using open source software. A larger company with 100 computers can save 6% on office software and 7% on server software.”[296]

Another research conducted by TheOpenEnterprise.com points out that 74% of the corporate managers interviewed consider lower costs as the main benefit from the usage of Open Source and Open standards-based software (see Figure 27 below. And 66% of the same group of managers consider the cost of Open Source software

[295] Source: bibliography 78
[296] This study included 50 large German companies and organizations, as well as 30 software retailers. Source: bibliography 102.

to be between 25% and 75% lower than proprietary alternatives (See Figure 28 below).

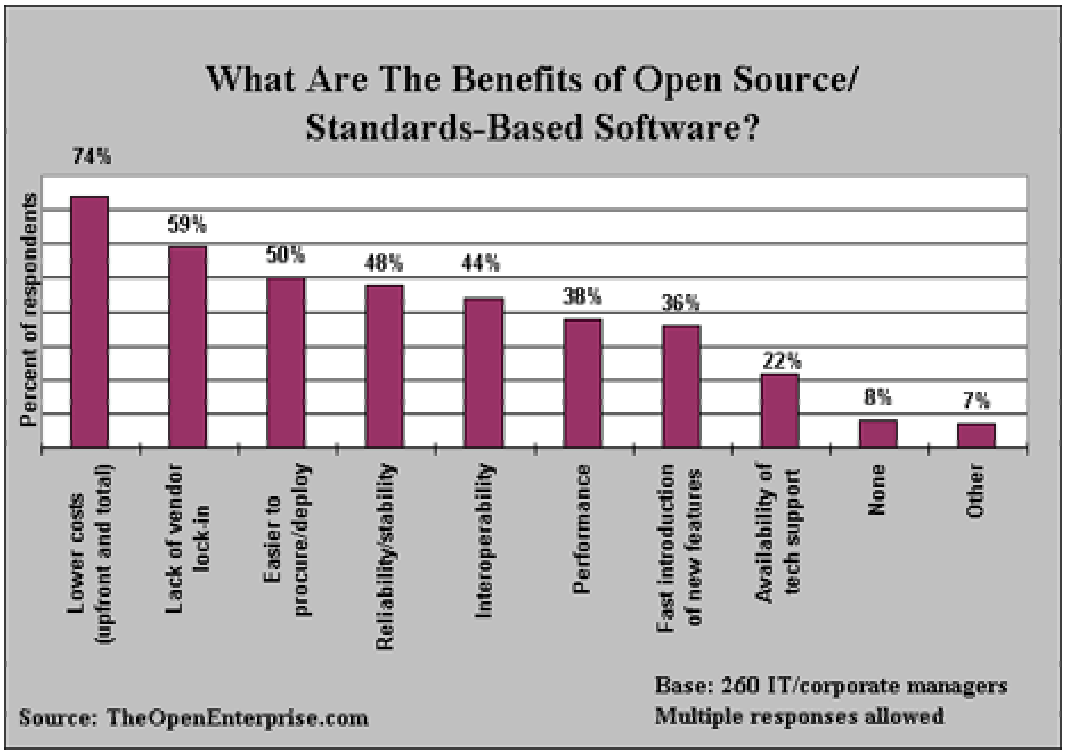


Figure 27 – Benefits of Open Source / Standards-based software

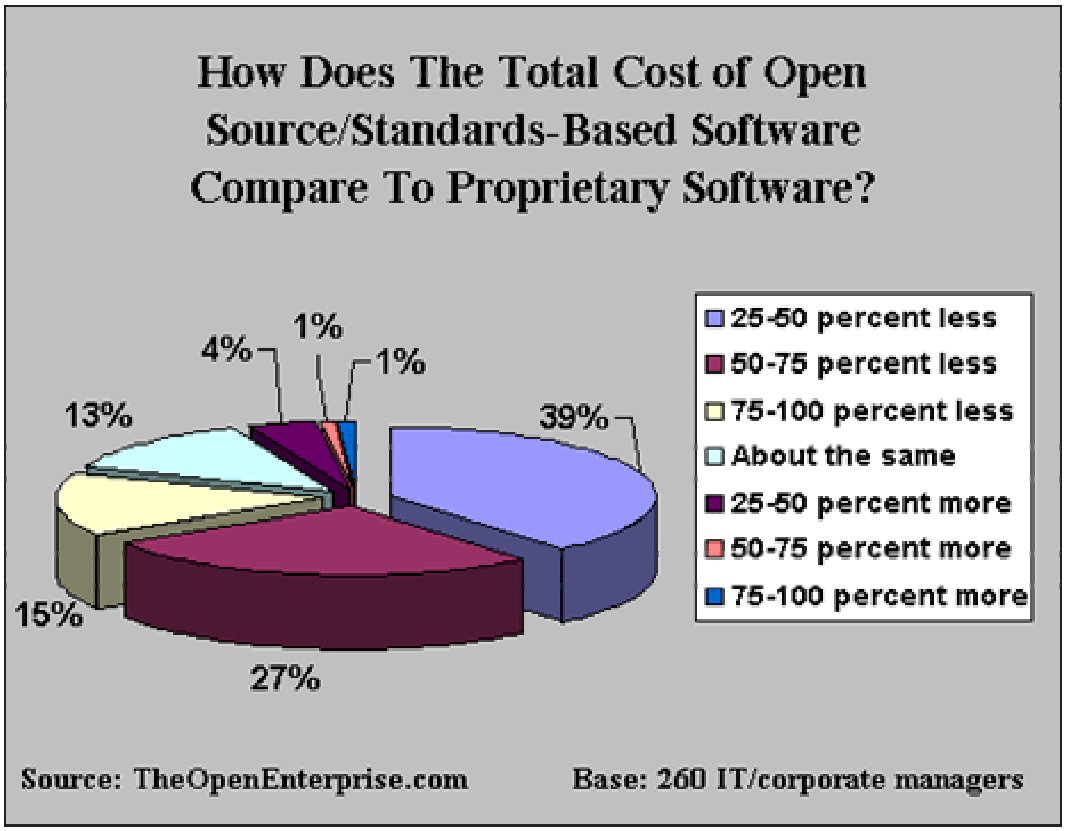


Figure 28 – Comparison between Open Source and Proprietary software costs

By opposition to the above cases, studies conducted by IDC research show that in some cases proprietary solutions like Windows may appear cheaper – in the long run – than Open Source alternatives. This study shows that Linux is certainly best for web sites hosting but for other server applications, Windows may be cheaper, due to reduced training costs. This study should be considered cautiously as it has been sponsored by Microsoft, by interviewing 104 American IT managers, related to the implementation of printer, security and file servers. One of the most important cost differences was for a security server, which would cost USD 91.000 under Linux compared to USD 70.000 for Windows (over 5 years). The main fact from the study is the low participation of license costs in the total implementation of the system (around 5%)[297].

6.5. Evolution and Control: Two flavours, four strategies

Considering the elements discussed in the previous sections - and based on the conclusions by Shapiro and Varian - four different strategies may be adopted by the information suppliers (like software and hardware vendors and internet service providers) when creating a technological paradigm. One criterion is the adoption of open or closed standards (control); the second is the option to create a new revolutionary technology or to keep the evolution of former technologies[298]. These factors shall be recognised when implementing a technological change, to better know the changing costs in advance, and to avoid lock-in situations. This section concentrates in the analysis of the suppliers viewpoint.

6.5.1. Openness or Control

Proprietary architectures may give enormous benefits for the network enterprise built around the companies controlling them, when the positive feedback is generated. The controlling companies are the copyright owners and the producers of the core technology. The control is kept when those companies believe their products to be widely accepted without the help from certification and standard organisations, and start to build a supplier and distribution network around their technologies. Normally commercial coalitions are formed quicker than standards are approved, so their chance to trigger a feedback effect is higher if the users are convinced of their technological superiority. This has been the reasoning publicly defended by Microsoft and Intel.

The major benefit for the controlling companies is when the network value becomes higher than the competition. The feedback and lock-in effects are created and the tendency for the network is only to grow until a new paradigm arrives or a complete monopoly is created. Due to the feedback, new users are attracted to profit from the high network value; due to the lock-in, existing users have problems to change of technology.

[297] Source : Bibliography 83
[298] Source: bibliography 7, page 182.

6.5.2. Performance or Compatibility

When implementing a new technology to replace the current generation, two options exist: to ensure a backward compatibility – evolution -, or to create new functionalities and a performance level that can convince the users to abandon their current product in favour of the new one - revolution.

A compromise may sometimes be found and it's by far the best option. Guarantying the compatibility while giving more performance and functionalities will reduce the changing costs – a precious argument to gather users from the previous technology – while being attractive to new users.

This is the ideal situation and does not happen often in the information economy. What may facilitate the users to migrate to the new technology is the existence of conversion tools – to easily convert information in the old format – or the availability of bridges, in which the new product can read information created by the previous one. When the standards and formats are open, this is simply a question of development. However, closed standards may give the controlling company the right to legally avoid these features to exist, or to ask the payment of high license fees. This is again a reason to implement products based on open standards and formats: even if the customers don't have high costs to terminate a lock-in situation, the companies creating new technologies may be simply inhibited to innovate.

6.5.3. The strategies

	Control	Openness
Compatibility	Closed migration	Open Migration
Performance	Superiority by Performance	Disruption

Table 4 – Evolution and Control strategies

To summarize the concepts of this section, let us briefly describe the four strategies:

- **Closed Migration** – This is normally the release of new versions of a proprietary product, and should not present a risk (for the supplier or user) if the backward compatibility is fully guaranteed, and the migration tasks are not complicated enough to produce a high changing cost. Two examples of closed migration are the successful replacement of DOS by Windows, and the limited usage of Mac OS.

- **Superiority by performance** - This is the introduction of a new and proprietary technology, incompatible with the dominant market solution. The changing costs should be calculated upon the existence of bridges and conversion tools – tested before to ensure a complete compatibility of the totality of the user information. Attention should be made to the credibility of eventual announcements made by the company owning the current market solution, to avoid the "vapourware" to destroy innovation. A market analysis should also be done to verify the existence of an open alternative, with similar changing costs, but with higher control – for

the user. A thriving story is the implementation of the UNIX operating systems, by opposition to the PS/2 and OS/2 failure to conquer the PC market.

- **Open Migration** – This is the preferred choice for the customers, with the solution being offered by different hardware, software and service suppliers, reduced changing costs, reduced risk of lock-in by the new product. For the supplier (without the ambition to create a monopoly, and accepting the challenge represented by the existence of innovative competitors) this is also a comfortable situation, as it may benefit from a wider network – even with the help from the competition – that can bring benefits when launching future initiatives or products. The UNIX wars show that the success of this strategy is not always guaranteed, and the creation of the single UNIX specification is a good example of the perseverance needed to attain such a goal.

- **Discontinuity** – This happens when a new technology appears, and it's implemented by many suppliers. Its characteristics are similar to the previous strategy, with the exception of the existence of changing costs. It's also favourable to the most competitive suppliers, able to provide an added-value network of service and compatible hardware and software products. To exemplify, the triumphant implementation of the OSI standard and the Internet protocols and the flourishing Linux server market are good examples, by opposition to the difficulty of Linux to conquer desktop users, with the need to improve the compatibility bridges (like the Wine[299] software) and migration tools (like OpenOffice, that accepts documents created by Microsoft Office).

6.6. The shift of power

In the end of the 1960s - the beginning of the information age – the computer manufacturers were focused in the production of machines, which came coupled to the operating systems. As discussed in the beginning of this chapter, in the 1980s the surge of the UNIX systems and the TCP/IP-based computer networks allowed the companies to form strategic alliances becoming network enterprises. With the consequent need for the flexible, interactive manipulation of computers, software became the most dynamic segment of the industry.

Irony, it seems, is that again an operating system from the UNIX family is helping to accelerate a paradigm shift: from software to services. This time the network was also important. The computer networks worked as the media, but the social peer networks were the real catalysts.

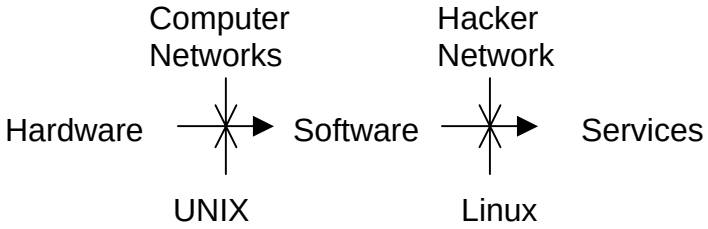


Figure 29 – UNIX and Linux catalysing the shift of informational power

[299] http://www.winehq.com/

This time IBM was one of the first big companies to react. Instead of considering Linux as another concurrent of its 5 operating systems (zVM, zOS, VSE, AS400 and AIX), IBM became part of the open source network, by sponsoring projects, creating laboratories around the world dedicated to open the door of every hardware platform and investing in Marketing[300]. Other major companies interested in promoting Linux are Intel, HP, SAP, SAS and Oracle. There are two (non-exclusive) possible reasons for this: the first is pure marketing, the second a potential service market around Linux. According to TheOpenEnterprise.com (See Figure 30 below), 43% of the companies may consider to implement enterprise applications.

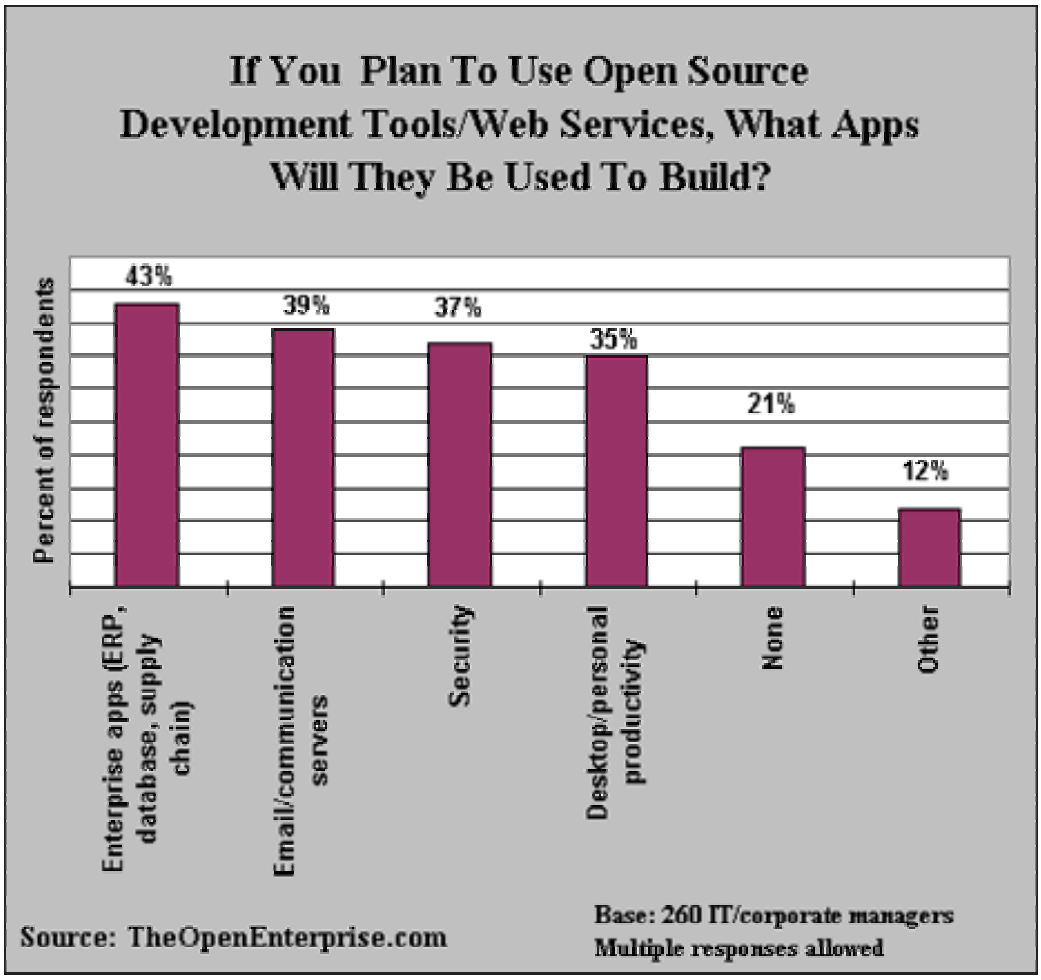


Figure 30 – Potential applications to be hosted in Open Source solutions

6.6.1. Behind the marketing scenes

HP, IBM and Intel are big players in the hardware arena. Today they work together with proprietary and open systems. In the case Linux growth previsions are true, and it finally dominates the server market in the near future, their interest is to be part of the new and powerful Linux network as hardware suppliers, keeping their market share.

[300] IBM won the first prize [of the Linux media awards, organized by Linux Media AG, with a jury of authors, developers and leading members of the Open Source community] as the company who has done the most to promote Linux during the year of 2002. source: Linux Magazine #25, November 2002.

These companies will keep their investments in the current operating systems, and will continue to actively participate in the Windows network. If the paradigm does not happen, due to important changing costs - in a period of slim IT budgets -, and Linux keeps a low part of the operating systems market, they have not lost their investment. They will enter to history as Linux Maecenas, have a good image to the open source community, and finally remain active in its market share.

6.6.2. The experience economy[301]

Let us consider one statement from Castells' analysis of the classical theory of post-industrialism: "Economic activity would shift from goods production to services delivery. The demise of agricultural employment would be followed by the irreversible decline of manufacturing jobs, to the benefit of service jobs that would ultimately form the overwhelming proportion of employment. The more advanced an economy, the more its employment and its production would be focused on services"[302]. An extrapolation of this sentence would produce a complementary one: The more advanced a company, the more its research and development would be focused on the production of services. This is the strategy from IBM, followed by the other companies.

The spread of the services realm, by the replacement of internal labour forces did not happen as predicted by analysts like Rifkin[303]. The usage of highly specialized consultancy services increased during the years preceding the Y2K issue, the implementation of the Euro currency and the hype of Internet start-ups. They were long-term and expensive contracts. The crisis originated by the flop of the Internet bubble and accentuated by the spawn of terrorism and the menace of another gulf war changed the figures and projections. The current tendency is to keep the knowledge inside the enterprise, dedicated to satisfy the real business needs. The services are still required, in a smaller scale and for short-term assignments, and concentrated in the infrastructure. Stated another way, if knowledge is power, the enterprises prefer to keep it internally pushing the core business to increase profit, and to use external forces - specialized in the volatile technologies - to decrease cost.

The installation of a new hardware, software or network component is surrounded by several complementary activities – some already discussed in this document -, like capacity planning, installation, migration from former systems, training, consolidation of smaller servers into a large one, or into a cluster. The main advantages of Linux and open standards for service providers are:

- The sharing of R&D costs with the other members of the open source network;
- The availability of the source and protocols for problem investigation and a better understanding of the logic behind the software;
- The easier communication across different hardware and applications;
- The pervasiveness of Linux - able to run in any existing platform, and quickly adaptable to eventual new ones - thus giving the choice to the service provider to

[301] The study in this section is empirically based in my previous essay (e*conomy) and in my personal observations and experience in the mainframe services market in the last decade.
[302] In bibliography 1, page 218.
[303] See bibliography 32.

recommend hardware equipments from best partners, and to change these partners without compromising its recommendations, and reduced need of retraining.

This motivates the companies to participate in the standard committees, to be aware before the technological changes, but mainly to create lobbies to influence decisions, favouring technologies implemented by their coalitions, and blocking standards with new innovations from the competition.

6.6.3. Branding

Open source software is a commodity market. In any commodity market, customers value a brand they can trust. The brand building, in open source, is highly based in supporting the community. Marketing strategies from the big companies joining the open source movement (among them IBM, Intel and Oracle) consist in announcing their investment in open source, by creating laboratories specialized in testing new versions of open source software with hardware components and helping to port open source operating systems to new hardware developments. Another common tactic is to sponsor developers to produce open source code.

On the other hand, Microsoft has a hard work to change the bad image associated with its brands. David Stutz – The man formerly responsible for Microsoft's anti-open source strategy – said: "Recovering from current external perceptions of Microsoft as a paranoid, untrustworthy, greedy, petty and politically inept organisation will take years."[304]

[304] Source: www.synthesist.net

7. Politics

"Technology does not determine society. Nor does society script the course of technological change. (…) Yet, it can, mainly through the state, suffocate its development." Manuel Castells

In the previous sections, we discussed the social and economical viewpoints of the adoption of open standards and open source. The provisory conclusions are the need for open standards to guarantee the freedom of choice and a healthy market competition, and the advantages of open source code to guarantee the innovative pace of technology with benefits for the privacy and security. Now we will analyse the political aspects: How can the European governments and the European Community – in the scope of this analysis – recognise these needs, and how can them support this openness?

Let us base the analysis in Castells' opinion: "firms do not seek technology for the betterment of humankind, but for profit and growth of value of their stocks. And political institutions are normally oriented toward maximizing the competitiveness of their constituent economies. Thus, profitability and competitiveness are the actual determinants of technological innovation and productivity growth."[305]

7.1. Openness by research

The academic environment is the natural root of the open standards, allowing the research initiatives to spread over different institutions and countries. As already discussed, this happened during the development of the UNIX operating system[306] and the Internet Protocols[307]. In opposition to commercial research – oriented by the market potentials of new products and technological features – academic researches are normally oriented towards innovation and quality – even when economic subsidies come from major companies like IBM. The political subsidies, whenever they exist, should stimulate this openness to continue in the centre of the research, and guarantee the quality of new technologies, which shall target the increase of social welfare.

This does not mean the suppression of proprietary architectures and closed software from the academic world. These technologies should continue to be investigated, and open bridges should be built between proprietary alternatives and Open Source. One of the possible solutions of the Microsoft antitrust case[308] would be to allow selected academic institutions to have access to selected parts of the source of selected Microsoft products. This selectivity should be forced by political means to be increased to the whole bunch of software code responsible for the interfaces between the Microsoft operating systems and Microsoft products (e.g. Windows and Internet Explorer) and in parts of the code where potential security or privacy breaches may put the social or private security under threat by commercial companies or crackers. Academic research could then be conducted to help to avoid

[305] In bibliography 1, pg 94
[306] See chapter 2.2.2
[307] See chapter 2.3.3
[308] Still under discussion when this document is being written.

potential risks, where Microsoft already proved unable (or unwitting or even unwilling) to act effectively.

The main objective of research in open technology, however, should be to guarantee the open software to evolve and be able to compete with commercial software in the same level. This happen by the technological improvements made by academic research, and by teaching the usage of this software to the students.

7.2. Openness by usage

The wide usage of Open Source software and the increasing usage of open standards by commercial software may be stimulated by letting the students use, learn and disseminate this knowledge in their future working environments. As already discussed, the license costs may have a low importance in the total implementation of the system (around 5%), and the main part comes from training[309]. The companies are more open to the implementation of pieces of software that are well known by current and future employees, as this may reduce the training costs of the technological departments and in the user community. This is the role of governmental institutions behind academic subsidies: wherever as possible, open software should be preferred to close and proprietary alternatives. Beyond the academic institutions, public entities could follow the same path.

The increasing importance of institutional projects in the IT market, in domains like e-government, education and health, is giving an important impulse in the usage of Open Source. As discussed before, mixed solutions integrating proprietary software and Open Source shall be considered, to guarantee the balance between the existing and new applications. This enforces the need for Open Standards, to be able to integrate both solutions, and to be able to easily migrate to new alternatives when there are economical or functional advantages.[310]

The American government is highly influenced by the major IT players like Microsoft, Intel and CISCO, who lobby against Open Source[311]. One of the arguments is the risk of reducing the profit – and consequently employment – of big companies part of ISC.[312] The increase in the usage of Open Source software in Europe does not have any negative impact in the economy, as the owners of main proprietary software are not European companies. In the opposite, they may benefit the service companies, by reducing the software license costs and allowing more projects to be implemented.[313] The European institutions and governments start to take advantage of this fact by using Open Source software in internal projects, and if the e-government initiatives really take shape in the next years, this may bring a real advantage for the open platforms.

[309] Source : Bibliography 83
[310] Source: Datanews n°05, 7/02/2003, page 16
[311] Source: bibliography 80
[312] In December 2002, after a research indicating that Open Source was widely used in the department of defense, this group of companies (united under the name ISC – Initiative for Software Choice") recommended the usage of proprietary solutions to the American government. Source: Datanews n°38, 6/12/2003, Page 3.
[313] It's a fact today that the budget restrictions are the main reason for projects to be cancelled or postponed. In 2002, lack of budget was behind 96% of the postponed projects in small and medium Belgian companies. Source: Datanews n°40, 20/12/2003, Page 8.

As an example, after some Italian and Brazilian regions, the Region of Brussels (COCOF[314]) voted in February 2002 a proposal to impose the usage of Open Source software. The usage of proprietary solutions may only be authorized in areas where no open alternative exists, with the planned period for the full migration from proprietary to Open Source software being only three years. Agoria[315], a Belgian entity representing many software suppliers, considered that legal interference from the government in technological domains should not happen.[316] During the writing of this document, ISC, the same group that is responsible for lobbying the American department of defense, is also pushing the Brussels government against such a decision. The arguments used by the ISC are purely juridical, and are quite incoherent with the practices of some companies in the very heart of the association (see below).

Another good example comes from the Belgian government, which voted on the 7th April 2003 a law to increase the transparency of electronic vote. The source code[317] is available on the internet.[318]

7.3. Openness by law enforcement

"We need a clear citizens' vision of the way the Net ought to grow, a firm idea of the kind of media environment we would like to see in the future. If we do not develop such a vision for ourselves, the future will be shaped for us by large commercial and political power holders." Rheingold

The most famous cases are always around Microsoft and the antitrust processes, run by the American and European governments. As this process has been extensively discussed[319], we will rather analyse the ISC arguments against the usage of open source by the Belgian governmental institutions.[320] We will analyze each of the arguments and show how can have an opposite interpretation and become a clear pro-Open Source argumentation.

The first "Nefarious" situation would be the risk for those institutions to be restricted to a single type of software systems. This is really a risk, but far higher for proprietary solutions with closed file formats and protocols, and this is never mentioned by ISC. In the future, if the institutions would like to migrate to newer software solutions, the conversion of existing information and adaptation of technical interfaces could easily be performed, with the help of documentation from the internet or the source code.

The second risk appointed by the ISC is the economical impact to the "Belgian Software Industry". Considering that most of the profit made by pure Belgian

[314] COmmission COmmunautaire Française - http://www.cocof.be/
[315] Covering a wide range of industrial sectors, Agoria represents a substantial portion of the Belgian economy: the federation's 1,200-plus member companies account for nearly one third of all Belgian exports of goods. Agoria's mission is to defend its members' interests as fully as possible and to bring all its influence to bear to improve the socio-economic environment in which they do business. http://www.agoria.be/
[316] Source: Datanews n°07, 21/02/2003, page 1
[317] In theory, the source code of the software Digivote and Jites may be found on the site http://elections.fgov.be/Nouveau/NouveauFr/Docunfr/codessources/Cdocu7nfr.htm. On the 5th May nothing was yet available …
[318] Source: Datanews n°16, 02/05/2003, page 1
[319] A good article may be found in page 1, Datanews n°02, 17/01/2003.
[320] Source: Bibliography 81

companies comes from service, and service is always needed, independent of the origin of the software, this argument is difficult to be understood.

A third argument, purely juridical, is that actions forcing institutions to use Open Source software would be against OMC dispositions to ensure the freedom of international commerce. Without entering in a juridical debate, we can easily challenge this viewpoint by arguing the validity of agreements forcing hardware suppliers to sell PCs with built-in Windows systems, with an implicit and mandatory cost for the end-user.

The fourth ISC argument is that the software market is dynamic and competitive enough to be self-ruled by commercial and technological differences. This is a very valid argument that could be used against the public known fact that companies making part of ISC are blocking the usage of Open Source solutions by building commercial alliances.

The fifth argument, which comes from the ISC objectives, is to guarantee that all pieces of software can participate equally in the market, without preferences or legal restrictions. We should wait for the legal conclusions of the antitrust cases against Microsoft, to see if legal restrictions are very independent of commercial lobbies.

7.4. Openness by stimulation

"Whatever chance remains for the survival of anything good may be in the preservation and availability of information, the only commodity that will be cheaper and more convenient." Ted Nelson

We discussed before that a mix between proprietary and open source alternatives may be preferred to a monopoly situation from either side. Many companies are highly influenced by mutual agreements, and ban open source solutions completely from their IT environments. To compensate this fact, governmental institutions could favour Open Source solutions and force the exchange of information with other companies to happen by open protocols – like XML – instead of proprietary standards – like MS-Word or MS-Excel spreadsheets. This can happen in e-government domains like taxes and social security, starting to be implemented today, and where conversions of software and file formats are already taking place. Considering the perspectives of increasing demand by such services, they may boom the open standards and open source adherence by the companies, which can later expand the usage of such technologies in internal applications.

In Belgium, the first phase of a project to implement tax payment online, with the development of 500 intelligent forms – a third of them oriented to the general public and companies, two thirds for internal usage – has chosen XML as the format for data exchange since 2001. This may give an impulse in the usage of XML in other companies and citizens, for other applications.

Part III – CODA

8. Open Future

"Although there are those among us now who have been granted the gift of being able to glimpse patterns of the future, probabilities tossed like dice on the uneven blanket of space and time, even these gifted ones know that no single future has been preordained for us or our posterity. Events are fluid. The future is like smoke from a burning forest, waiting for the wind of specific events and personal courage to blow the sparks and embers of reality this way or that." – Dan Simmons in The Rise of Endymion

The roles of desktops and servers will continue, but the peer-to-peer concept will allow them to act together, with groups of desktops joining forces to outperform powerful servers. This concept will continue to propagate to several domains and communities outside technology, with distant people gathering via the networks to make face to big commercial monopolies, and dominant political parties. Democracy rules will need to be reviewed, as the need for representatives will decrease, people being able to give their own opinion at any time or any place, and have enough free information to base their decision.

The real TCO - Total Cost of Ownership – is extremely difficult to be calculated, for implementations of complex applications that require intensive computing capacity, high availability, large databases, and communication with many users via the Net. Of course, it must be considered, but it's only one of the elements of a good analysis matrix including factors like usability, compatibility with existing systems, potential evolution of the hardware and software platform, and quality of support. Even if specialists, consultants and service providers are there to provide the good answers, a general knowledge about the infrastructure may increase the quality of the project definition, and consequently reduce the risk factor in the decision making process.

As said by Rheingold, "More people must learn about that leverage and learn to use it, while we still have the freedom to do so, if it is to live up to its potential. The odds are always good that big power and big money will find a way to control access to virtual communities; big power and big money always found ways to control new communications media when they emerged in the past. The Net is still out of control in fundamental ways, but it might not stay that way for long. What we know and do now is important because it is still possible for people around the world to make sure this new sphere of vital human discourse remains open to the citizens of the planet before the political and economic big boys seize it, censor it, meter it, and sell it back to us."

Not all software has to be open-source, though. Only that software that is critical to research, to security, or to ensure communication with other companies has to be open-source. Open protocols and open formats should be used essentially in critical areas, where the information must be always accessible independently of the usage of specific pieces of software. It's very reasonable for everything else to be proprietary and to then compete on that basis in the marketplace. It's how companies like HP, Oracle, Microsoft and IBM makes billions in revenue. It's how companies and application developers build their infrastructure. In fact, the better all the pieces – open source and proprietary – are integrated, the more the industry advances and the better everybody can compete with their proprietary software, applications, services, and all the things that drive the IT economy.

The usage of alternatives like MDA and XML may reduce the negative impact of proprietary alternatives, by virtually eliminating the risk of imprisonment in closed platforms and formats. However, they are not the only elements to be considered – as demonstrated in the part 2 of this study – and their usage must be clear and complete to be a real advantage. If the software platforms are built around XML-based documents, this solution must be implemented in every important part of the software, to hold all the information. No piece of closed format should be used for strategic data.

The social and economical advantages for most countries and small and medium companies should really push the governments to take part on this battlefield. As proven in the recent gulf war, economical and political fights are more important and take longer than the military ones. However, as stated by Himanen, “The hacker open model could be transformed into a social model. (…) We have seen that the hacker model can bring about great things in cyberspace without governments and corporations as mediators. It remains to be seen what great things individuals’ direct cooperation will accomplish in our ‘flesh reality’ ”[321].

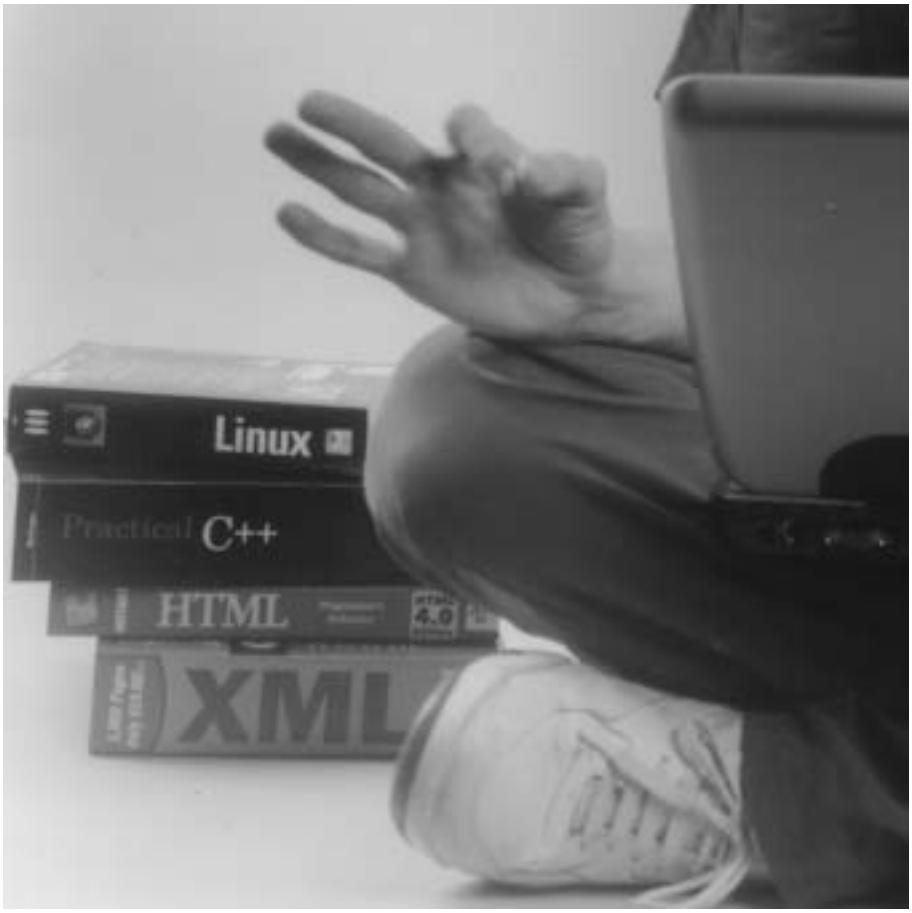
8.1. INCA[322]

Based on this study, a future landscape of the technical infrastructure has been suggested and is available online (http://www.k-binder.be/INCA/). After browsing the online chart, the reader can come back to the document, read selected topics and leave comments in the website.

[321] See bibliography 4, page 81.

[322] INCA stands for Intelligent and Net-Centric Architecture. I coined this term to define a possible way of connecting all technologies described on the previous chapters, and a possible evolution of the infrastructure platform – if implemented by open-minded spirits. As its name suggests, the network will replace the processor, being the heart of the architecture.

Part IV – Annexes



Appendix A. The Open book

This thesis is published online to allow Internet readers to read, comment and complement the theories - the final part of the conclusion is only available in the online version. The complexity of the document structure, with internal hyperlinks, did not allow the creation of a simple HTML document. For the moment, a web site has been created using LAMP to allow the readers to comment the overall work and form an online community. PDF has been chosen as a format for the document visualization.

The proprietary format will give place to XML, and LAMP (Linux/ Apache/ MySQL/ PHP) will be used in the technical infrastructure for its visualization. The online version will also contain hyperlink references to the products, brands, books and sites discussed in this document.

In parallel, to revise the content, newsgroups will be used to validate, and complement the concepts and theories.

To access the online version, go to www.k-binder.be and select “papers - thesis”.

Appendix B. Standards Organizations

Some of the most important organizations that participate with standard definitions, studies, analysis and control of the Internet protocols, names and addresses are[323]:

- World Wide Web Consortium (W3C)[324] – Created in October 1994, the W3C develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential. W3C is a forum for information, commerce, communication, and collective understanding.

- The Internet Engineering Task Force (IETF)[325] – The Internet Engineering Task Force (IETF) is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. It is open to any interested individual.

- Internet Activities Board (IAB)[326] – Created in September 1984, IAB is a technical advisory group of the Internet Society, who discuss issues pertinent to the Internet and set Internet policies through decisions and task forces. The IAB designates some Request For Comments (RFC) documents as Internet standards, including Transmission Control Protocol/Internet Protocol (TCP/IP) and the Simple Network Management Protocol (SNMP). The IAB provides advice to the IESG on working group formation and the architectural implications of the IETF working group efforts.

- Internet Engineering Steering Group (IESG)[327] – The IESG is the standards approval board for the IETF. It is responsible for technical management of IETF activities and the Internet standards process, administering the process according to the rules and procedures that have been ratified by the ISOC Trustees. The IESG ratifies or corrects the output from the IETF's Working Groups, gets WGs started and finished, and makes sure that non-WG drafts that are about to become RFCs are correct.

- The Internet Research Task Force (IRTF)[328] – Its mission is to promote research of importance to the evolution of the future Internet by creating focused, long-term and small Research Groups working on topics related to Internet protocols, applications, architecture and technology. The Research Groups are expected to have the stable long term (with respect to the lifetime of the Research Group) membership needed to promote the development of research collaboration and teamwork in exploring research issues. Participation is by individual contributors, rather than by representatives of organizations.

---

[323] Sources: bibliography 13,27 and 46, RFC 3160 (http://www.iesg.org/tao.html), RFC 2026 (http://www.iesg.org/rfc/rfc2026.txt).
[324] http://www.w3.org/
[325] http://www.ietf.org/
[326] http://www.iab.org/
[327] http://www.iesg.org/iesg.html
[328] http://www.irtf.org/. See also ftp://ftp.isi.edu/in-notes/rfc2014.txt

- Internet Society (ISOC)[329] - Brought into existence in January 1992, the Internet Society is a professional membership organization of Internet experts that comments on policies and practices additionally to overseeing a number of other boards and task forces dealing with network policy issues.

- Internet Assigned Numbers Authority (IANA)[330] – Based at ICANN, IANA is in charge of all "unique parameters" on the Internet, including IP (Internet Protocol) addresses.

- The Internet Corporation for Assigned Names and Numbers (ICANN)[331] – ICANN is the non-profit corporation that was formed to assume responsibility for the IP address space allocation, protocol parameter assignment, domain name system management, and root server system management functions.

- International Organization for Standardization (ISO)[332] — ISO is an international standards organization responsible for a wide range of standards, including many that are relevant to networking. Its best-known contribution is the development of the OSI reference model and the OSI protocol suite.

- American National Standards Institute (ANSI)[333] – Founded in October 1918, ANSI - which is also a member of the ISO - is the coordinating body for voluntary standards groups within the United States. ANSI developed the Fiber Distributed Data Interface (FDDI) and other communications standards.

- European Telecommunications Standards Institute (ETSI)[334] – Created in 1989, ETSI plays a major role in developing a wide range of standards and other technical documentation as Europe's contribution to worldwide standardization in telecommunications, broadcasting and information technology. Its prime objective is to support global harmonization by providing a forum in which all the key players can contribute actively. It tries to compensate ANSI's influence and it developed the standards Euro-RNIS and GSM.

- Electronic Industries Association (EIA)—EIA specifies electrical transmission standards, including those used in networking. The EIA developed the widely used EIA/TIA-232 standard (formerly known as RS-232).

- Institute of Electrical and Electronic Engineers (IEEE)[335] – Founded in 1963, the IEEE is a non-profit, technical professional association of more than 377,000 individual members in 150 countries, which defines networking and other standards. The IEEE developed the widely used LAN standards IEEE 802.3 and IEEE 802.5.

[329] http://www.isoc.org/
[330] http://www.iana.org/
[331] http://www.icann.org/
[332] http://www.iso.org/
[333] http://www.ansi.org/
[334] http://www.etsi.org/
[335] http://www.ieee.org/

- International Telecommunication Union (ITU)[336] – Founded on 17 May 1865, the ITU – headquartered in Geneva, Switzerland – is an international organization within the United Nations System where governments and the private sector coordinate global telecom networks and services.

- ITU Telecommunication Standardization Sector (ITU-T)[337] – It is one of the three Sectors of the International Telecommunication Union (ITU). It was created on 1 March 1993, replacing the former International Telegraph and Telephone Consultative Committee (CCITT) whose origins go back to 1865. Its mission is to ensure an efficient and on-time production of high quality standards covering all fields of telecommunications.

- Object Management Group (OMG)[338] – The OMG was founded in April 1989 by eleven companies[339] and in October 1989, it began independent operations as a not-for-profit corporation, including in 2002 about 800 members. The OMG was formed to create a component-based software marketplace by hastening the introduction of standardized object software. The organization's charter includes the establishment of industry guidelines and detailed object management specifications to provide a common framework for application development.

[336] http://www.itu.int/
[337] http://www.itu.int/ITU-T/
[338] http://www.omg.org/
[339] Including 3Com Corporation, American Airlines, Canon, Inc., Data General, Hewlett-Packard, Philips Telecommunications N.V., Sun Microsystems and Unisys Corporation

Appendix C. References

C.1. Trademarks

All trademarks are the property of their respective owners.

All the photos are © 2002 Joyce Binder.

C.2. Figures

C.3. Tables

Appendix D. Bibliography

Books

1. The Information Age: Economy, Society and Culture
   Volumes I – The Rise of the Network Society
   © 1996,2000 Manuel Castells
   Blackwell Publishers Ltd
   http://sociology.berkeley.edu/faculty/castells/

2. The Information Age: Economy, Society and Culture
   Volume II – The Power of Identity
   © 1996,2000 Manuel Castells
   Blackwell Publishers Ltd

3. The Information Age: Economy, Society and Culture
   Volume III – End of Millenium
   © 1996,2000 Manuel Castells
   Blackwell Publishers Ltd

4. The Hacker Ethic and the Spirit of the Information Age
   © 2001 Pekka Himanen
   Prologue © 2001 Linus Torvalds
   Epilogue © 2001 Manuel Castells
   Random House, Inc
   ISBN 0-375-50566-0
   http://www.hackerethic.org

5. Open Sources – Voices from the Open Source Revolution
   Chris DiBona, Sam Ockman & Mark Stone (many authors)
   © 1999 O'Reilly & Associates
   ISBN 1-56592-582-3
   http://www.oreilly.com/catalog/opensources/book/toc.html

6. The Cathedral and the Bazaar
   © Eric Raymond
   http://www.tuxedo.org/~esr/writings/cathedral-bazaar/
   (Online version)

7. Economie de l'information
   French translation of: Information Rules, 1st Edition
   © 1998 Carl Shapiro & Hal Varian
   Harvard Business School Press
   http://www.inforules.com/

8. Qu'est-ce que le virtuel
   Pierre Levy
   © 1998 La découverte
   ISBN 2-7071-2835-X
   http://hypermedia.univ-paris8.fr/pierre/virtuel/virt0.htm

9. L'intelligence collective : pour une anthropologie du cyberspace
   Pierre Levy
   © 1997 La découverte
   ISBN 2-7071-2693-4

10. The Age of Spiritual Machines
    © 1999 Ray Kurzweil
    Ed. Phoenix
    http://www.kurzweilai.net/meme/frame.html?main=/articles/art0281.html
    ISBN 0-75380-767-X

11. MySQL et PHP
    Philippe Rigaux
    © 2001 Éditions O'Reilly – Paris
    http://www.oreilly.fr/catalogue/mysql_php.html
    ISBN 2-84177-123-7

12. UML Specification
    OMG
    http://www.omg.org/technology/documents/formal/uml.htm

13. Internet-Enabled Business Intelligence
    William A. Giovinazzo
    © 2003 Pearson Education, Inc
    ISBN 0-13-040951-0
    http://vig.pearsoned.com/store/product/1,3498,store-562_isbn-0130409510,00.html

14. Exploring IBM zSeries and S/390 Servers
    Jim Hoskins and Bob Frank
    Seventh edition
    © 2002 Maximum Press
    ISBN 1-885068-89-1
    http://www.maxpress.com/catalog/mainframes.html

15. Operating Systems Handbook
    © 2001 Bob DuCharme
    ISBN 0-07-017891-7
    http://www.snee.com/bob/opsys.html

16. UNIX as a second language
    Bob Johnson
    ISBN 0-9650929-1-7

17. The Road Ahead
Bill Gates
ISBN 0-670-77289-5
http://www.roadahead.com/

18. Le hold-up planétaire: La face cachée de Microsoft
Roberto Di Cosmo / Dominique Nora
ISBN 2-7021-2923-4
http://www.pps.jussieu.fr/~dicosmo/HoldUp/

19. Instant HTML
Alex Homer, Chris Ullman and Steve Wrigh
© 1997 Wrox Press
ISBN 1-861001-56-8
http://www.wrox.com/books/1861001568.htm

20. The Virtual Community
©1998 Howard Rheingold
http://www.rheingold.com/vc/book/intro.html
(Online version)

21. Sexe, Mensonges et Internet
Yves Thiran
© 2000 Cordon Art
ISBN 2-8040-1493-2
http://www.labor.be

22. Internet, et après ?
Dominique Wolton
© 2000 Flammarion
ISBN 2-08-081459-1

23. L'imposture informatique
François de Closets / Bruno Lussato
© 2000 Librairie Arthème Fayard
ISBN 2-213-60849-0

24. Computer Science : an overview
J.Glenn Brookshear
© 1997 Addison Wesley Longman, Inc
ISBN 0-8053-4632-5

25. XML Complete
Richard Mills and Tom Cirtin
© 2001 SYBEX, Inc
ISBN 0-7821-4033-5
http://www.sybex.com/

Papers

26. Software Development: Past, Present and Future - Trends and Tools
    Patrick Gerland
    http://www.ons.dz/unfpa/papers/nidi_pg.pdf

27. Technologies du multimédia, des télécommunications et de l'Internet
    [version 7.57]
    Université de Liège - Institut d'Electricité Montefiore
    Prof. Marc Van Droogenbroeck
    http://www.ulg.ac.be/telecom/multimedia/dir7325/total-multimedia.pdf

28. Information paper - Internet Domain Name System Basics
    ITU
    Document INF/6-E - 14 October 2002
    http://www.itu.int/osg/spu/mina/2002/inf6-E.html

29. XML in practice
    © 2000 Xephon plc

30. The XML files
    Aaron Skonnard
    © 2002 Microsoft Corporation

Essays

31. Peer to peer: from technology to politics to a new civilisation?
    Michel Bauwens

32. e*conomy – The Experience economy
    http://www.k-binder.be/Papers/
    Jean Binder

Articles

33. Datanews
    2002: numbers x
    2003:

34. WIRED - January 2003
    Spectrum wants to be free (page 082)
    Kevin Werbach

35. WIRED - January 2003
    Linux for the Wal-Mart Crowd (page 085)
    Michael Robertson

36. Financial Times
    January 27 2003
    IBM push on grid computing
    Fiona Harvey

Internet

37. The Evolution of High-End Servers
http://www.esj.com/features/article.asp?EditorialsID=120

38. The Evolution of the Unix Time-sharing System
Dennis M. Ritchie - Bell Laboratories
http://cm.bell-labs.com/cm/cs/who/dmr/hist.html

39. Perspective: Moving beyond creative cloning
http://news.com.com/2010-1071-965752.html

40. BSD Operating Systems: Perspective
http://www.gartner.com/DisplayDocument?id=308056&ref=g_search

41. A Condensed History of Personal Computing
http://www.landiss.com/history.htm

42. Windows Desktop Operating Systems
http://www.microsoft.com/windows/winhistorydesktop.mspx

43. Windows Server Operating Systems
http://www.microsoft.com/windows/winhistoryserver.mspx

44. Internet Protocol
http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/ip.htm

45. TCP/IP
CISCO
http://www.cisco.com/univercd/cc/td/doc/product/iaabu/centri4/user/scf4ap1.htm

46. Network Basics
CISCO
http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/introint.htm#xtocid5

47. Internetworking Technology Handbook
CISCO
http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/

48. DNS – Introduction
ITU / Nominum
http://www.itu.int/itudoc/itu-t/workshop/enum/011.html

49. Standards and Specifications List
IST - Information Society Technologies
http://www.diffuse.org/standards.html

50. RFC 1336 - Who's Who in the Internet
G. Malkin / Xylogics
http://www.iesg.org/rfc/rfc1336.txt

51. RFC 2026 - The Internet Standards Process - Revision 3
S. Bradner / Harvard University
http://www.iesg.org/rfc/rfc2026.txt

52. RFC 3160 – The TAO of IETF
S. Harris / Merit Network
http://www.iesg.org/tao.html

53. History of ARPANET
Michael Hauben
http://www.dei.isep.ipp.pt/docs/arpa.html

54. How the Internet Really Works
Vinton Cerf
http://www.netlingo.com/more/cerfart.html

55. Computer Networking: Global Infrastructure for the 21st Century
Vinton Cerf
http://www.cs.washington.edu/homes/lazowska/cra/networks.html

56. The Internet After the Fad
Remarks of Dr. Robert Metcalfe at the University of Virginia
May 30, 1996
http://americanhistory.si.edu/csr/comphist/montic/metcalfe.htm

57. Cramming more components onto integrated circuits
Gordon E. Moore, 1965
http://www.intel.com/research/silicon/moorespaper.pdf

58. The Third Place
Jeff Tidwell
http://www.infonortics.com/vc/1999/tidwell/sld001.htm

59. The future of networking
© BRIE 1993 - Research Paper by Michael Borrus and François Bar
http://brie.berkeley.edu/~briewww/pubs/rp/network.html

60. From Partial to Systemic Globalization: International Production Networks in the Electronics Industry
Dieter Ernst - April 1997
http://brie.berkeley.edu/~briewww/pubs/wp/wp98.html

61. The Matrix: Computer Networks and Conferencing Systems Worldwide
John S. Quarterman
© 1990 Digital Press
http://www.mids.org/books/matrix/

62. The Future of Networking
Michael Borrus and François Bar
© 1993 by BRIE - March 16, 1993
http://e-conomy.berkeley.edu/publications/wp/network.html

63. Ted Nelson and Xanadu
© 1993-2000 Christopher Keep, Tim McLaughlin, Robin Parmar
http://www.iath.virginia.edu/elab/hfl0155.html

64. The Identity Wars
Enterprise Systems
By John Harney - ASPWatch
http://esj.com/features/article.asp?EditorialsID=88

65. New computer chip: useful tool or privacy invasion?
Paul Van Slambrouck - The Christian Science Monitor
http://www.csmonitor.com/durable/1999/02/16/fp2s2-csm.shtml

66. Intel Nixes Chip-Tracking ID
Wired News - Declan McCullagh
http://www.wired.com/news/politics/0,1283,35950,00.html

67. Intel on Privacy: 'Whoops!'
Wired News - Polly Sprenger
http://www.wired.com/news/politics/0,1283,35950,00.html

68. HTTP State Management Mechanism
IETF / RFC 2109
http://www.iesg.org/rfc/rfc2109.txt

69. Worldwide Linux Operating Environments Forecast, 2002–2006: Client Shipments Pick Up the Pace
Analyst: Al Gillen
© IDC 2002
http://www.hp.com/united-states/linux/images/Linux_Operating_Forecast_2002-2006.pdf

70. Open Standards - Definition
National Library of Canada
http://www.nlc-bnc.ca/9/13/p13-103-e.html

71. Open Standard definition
The IT University of Copenhagen
http://linuxlab.dk/openstandards/

72. Deciphering the open-source war
Bruce Perens
Cnet News.com
http://news.com.com/2010-1078-855155.html

73. More Than a Gigabuck: Estimating GNU/Linux's Size
©2001 David A. Wheeler
http://www.dwheeler.com/sloc

74. Counting potatoes: The size of Debian 2.2
©2001 Jesús M. González-Barahona, Miguel A. Ortuño Pérez, Pedro de las Heras Quirós, José Centeno González and Vicente Matellán Olivera.
http://people.debian.org/~jgb/debian-counting/counting-potatoes/

75. Microsoft Windows 2000 Server to Linux Comparison
© 2001 Microsoft
http://members.microsoft.com/partner/products/Servers/Windows2000Server/Windows_2000_Server_to_Linux_Comparison.aspx

76. A strategic comparison of Windows vs. Unix
© 2001 LinuxWorld.com / Paul Murphy
http://www.linuxworld.com/site-stories/2001/1018.tco.html

77. Linux vs. Windows - Total Cost of Ownership Comparison
© 2002 Cybersource® Pty. Ltd.
http://www-1.ibm.com/linux/linuxvswindowstco.pdf

78. Unilever moves to Linux for savings
31-01-2003
Andy McCue.
http://www.vnunet.be/detalle.asp?ids=/News/Enterprise_Computing/Strategy/20030131023

79. Morgan Stanley turns to Linux
30-01-2003
Jonathan Collins
http://www.vnunet.be/detalle.asp?ids=/News/Enterprise_Computing/Strategy/20030130004

80. "Fixer une frontière à l'open source"
28-11-2002
Jose Delameilleure
http://www.vnunet.be/detalle.asp?ids=/News/Enterprise_Computing/20021128013

81. Un décret pro-logiciels libres jugé 'inopportun' et "néfaste"
25-03-2003
Olivier Fabes
http://www.vnunet.be/datanews/detalle.asp?ids=/News/Top_Stories/Enterprise_Computing/20030325006

82. Retired Microsoft man issues Linux warning
20-02-2003
Nick Farrell
http://www.vnunet.be/detalle.asp?ids=/News/Enterprise_Computing/Technology/20030220015

83. Linux plus cher que Windows
04-12-2002
Jose Delameilleure
http://www.vnunet.be/detalle.asp?ids=/News/Enterprise_Computing/20021204009

84. Understanding XML
Eric Armstrong
http://java.sun.com/webservices/docs/1.0/tutorial/doc/IntroXML.html

85. Clear-cut choices in battle over Web services
By Brad Murphy
Special to ZDNet - December 12, 2002
http://zdnet.com.com/2100-1107-977034.html

86. What's Next: The Future of Web Services
Microsoft
http://msdn.microsoft.com/webservices/understanding/whatsnext/default.aspx

87. What are XML Web Services?
Microsoft - January 14, 2002
http://www.microsoft.com/net/basics/xmlservices.asp

88. .NET Glossary
Microsoft - March 31, 2003
http://www.microsoft.com/net/basics/glossary.asp

89. How to Get .NET
7 Steps to Connecting Your World of Information, People, Systems, and Devices
Microsoft - September 13, 2002
http://www.microsoft.com/net/basics/glossary.asp

90. Defining the Basic Elements of .NET
Microsoft - January 24, 2003
http://www.microsoft.com/net/basics/whatis.asp

91. What Is .NET?
Microsoft
http://www.microsoft.com/net/basics/

92. Whatis: .NET
searchWebServices.com Definition
http://searchwebservices.techtarget.com/sDefinition/0,,sid26_gci342248,00.html

93. Microsoft® Host Integration Server 2000 Product Overview
© 2002 Microsoft Corporation
http://www.microsoft.com/hiserver/techinfo/HISoverviewWP.pdf

94. WEB SERVICES AND OPEN SOURCE
Preston Gralla
Part 1:
http://searchwebservices.techtarget.com/tip/0,289483,sid26_gci868183,00.html
Part 2:
http://searchwebservices.techtarget.com/tip/1,289483,sid26_gci870011,00.html?FromTaxonomy=%2Fpr%2F288973

95. LAMP Lights the Way to Web Services for Financial Reports Firm
08/02/2002 - Robert McMillan
http://www.oetrends.com/cgi-bin/page_display.cgi?75

96. XML and the Second-Generation Web
JON BOSAK and TIM BRAY - May 06, 1999
http://www.sciam.com/article.cfm?articleID=0008C786-91DB-1CD6-B4A8809EC588EEDF

97. Introduction to OMG's Unified Modeling Language™ (UML™)
© 1997-2002 Object Management Group, Inc.
http://www.omg.org/gettingstarted/what_is_uml.htm

98. UML 2001: A standardization Odissey
Cris Kobryn
© 1999 ACM 0002-0782/99/1000
(This article appeared in Communications of the ACM, vol. 42, no. 10, October 1999)
http://www.omg.org/attachments/pdf/UML_2001_CACM_Oct99_p29-Kobryn.pdf

99. MDA® Specifications
© 1997-2003 Object Management Group, Inc.
http://www.omg.org/mda/specs.htm

100. CORBA BASICS
© 1997-2003 Object Management Group, Inc.
http://www.omg.org/gettingstarted/corbafaq.htm

101. Extreme Programming and Open Source Software
Message Posted 13 Nov 2000 by “apm” and replies
http://www.advogato.org/article/202.html

102. Study: Big companies save big from open source
By John Blau
© IDG News Service, 05/08/03
http://www.nwfusion.com/news/2003/0508studybigc.html

Seminars & Commercial Presentations

103. Open Source and Free Software
Richard M. Stallman – GNU Project

104. e-mail evolution
By Eric Allman, author of sendmail

105. Intel and Linux
Rod O'Shea - EMEA Enterprise Business Director - Intel

106. SAS and Linux
Patrick Xhonneux - Director Business Development - SAS Institute

107. Oracle and Linux
Dries Cuypers - Product Marketing Manager - Oracle Belgium

108. SAP and Linux
Denis Rousseau - Business Development Manager - SAP AG

109. HP and Linux
Urs Rengli - Director Marketing and Alliances EMEA - HP

110. IBM and Linux
Several IBM speakers

111. High Availability Solutions for Linux
John Banfield - EMEA Director - Steeleye

Appendix E. Index

J

K

L

M

N

O

P

Q

R

S