

**MAHENDRA ARTS AND SCIENCE COLLEGE
(AUTONOMOUS)
DEPARTMENT OF COMPUTER SCIENCE & APPLICATIONS**

Paper code : M16UCAE03
Paper Name : Open Source Software Development
Class : III BCA
Semester : V

Presented by
P.Sangeetha, AP/CSA,MASC




SYLLABUS

- ▶ **UNIT – I**

- ▶ **Open Source Software: Definitions and History:-**Definition of Terms-A Brief History of Software-**Where Open Source Is Successful:** Analytical Framework-Open Source Is in Widespread Successful Use-**Open Source: The Good, the Bad and the Ugly:** What Is Good about Open Source-Open Source Is Not Enough by Itself-How Choosing Open Source Is More Difficult for You-What Others Say about Open Source.


What is Open Source Software Development ?

- Open-source software development is the process by which open-source software, or similar software whose source code is publicly available, is developed by an open-source software project. These are software products available with its source code under an open-source license to study, change, and improve its design.
 - Open source software is different. Its authors make its source code available to others who would like to view that code, copy it, learn from it, alter it, or share it.
 - LibreOffice and the GNU Image Manipulation Program are examples of open source software.
 - GIMP is a cross-platform image editor available for GNU/Linux, OS X, Windows and more operating systems.
- 

UNIT – I

OPEN SOURCE SOFTWARE: DEFINITIONS AND HISTORY

1. Definition of Terms

- Open source software is software that must be distributed with source code. Most of them download from the Internet.
 - The Open Source Definition (OSD) was originally written by Bruce Perens for Debian Linux and was completed in 1997. Open source that is simple without being too simple, and it includes several licenses that are acceptable.
 - Apache, or Mozilla licenses, called “Open Source Software.” or “Free Software.” In Europe, call it “Free/Libra or Open Source Software” (FLOSS).
- 


What Is Free Software?

- The term free software dates from 1984, published by Richard Stallman. The Free Software Web site is <http://www.gnu.org/>.

- The term open source dates from 1997, when a group of people, including Eric Raymond, Tim O'Reilly, and Bruce Perens, The open source Web site is <http://www.opensource.org>.

What Are Good Examples of Open Source?


Open source software include:

- The Apache Web server, with a share of 65 percent of installed world-wide Web servers.
 - The Linux operating system, used on millions of servers.
 - The GNU C/C++ language suite, used to build Linux and Apache and almost every operating system.
- 

Does “Open Source” Mean Linux?

- It is possible to adopt open source without using Linux at all.
- For example, Apache is an open source Web server that runs on Windows and UNIX servers and IBM mainframes as well as Linux.

Does Open Source Require Different Business Methods?

- It is not necessary to adopt an open source approach to development or distribution in open source software.
 - Many companies develop closed code software products that run on open source software and sell them for a profit. For example closed code software companies are IBM, HP, Oracle, SAP, and Apple.
- 

Will All Systems Be Open Source One Day?

There are advantages to open source but also limits.

- First, reasons for secrets.
- Second, reasons for complex business areas never really become open source.
- Third, reasons for new wave of innovation and offer.

Is Open Source a Fad That Will Go Away?

- The information technology business has seen its share of fads. Mature products such as office suites often become commodities



1.2 A BRIEF HISTORY OF SOFTWARE


Early Years

➤ In the 1950s, the first modern business computers were introduced. Custom programs are written in binary code.

➤ In the 1960s, commercial mainframes were introduced by several companies. System software was developed in assembler language. A major success of the System/360 was introduced by IBM in 1964.

UNIX

➤ UNIX is an operating system which was first developed in the 1960s. By operating system, we mean the suite of programs which make the computer work.




- It is a stable, multi-user, multi-tasking system for servers, desktops and laptops.
- UNIX systems also have a graphical user interface (GUI) similar to Microsoft Windows which provides an easy to use environment
- UNIX was written in C and constructed as a set of small programs that worked well together. By 1974, the UNIX system, including the C language and tools, had been ported to several platforms.

BSD

Berkeley Software Distribution

- The University of California at Berkeley had the UNIX Version 4 code in 1974.
- By 1975, Berkeley engineers, including Bill Joy, were adding tools and kernel improvements, and in 1977 a “Berkeley Software Distribution” (BSD) was formed. BSD was issued and licensed to 500 machines by 1980.

GNU

- In 1984, free software began with GNU. GNU is derived from the acronym “GNU’s Not UNIX,” which is recursive (on the G).
 - GNU was begun by Richard Stallman as a project to develop a complete operating system that was modeled on UNIX functionality and philosophy but that used no code from UNIX.
- 

Linux


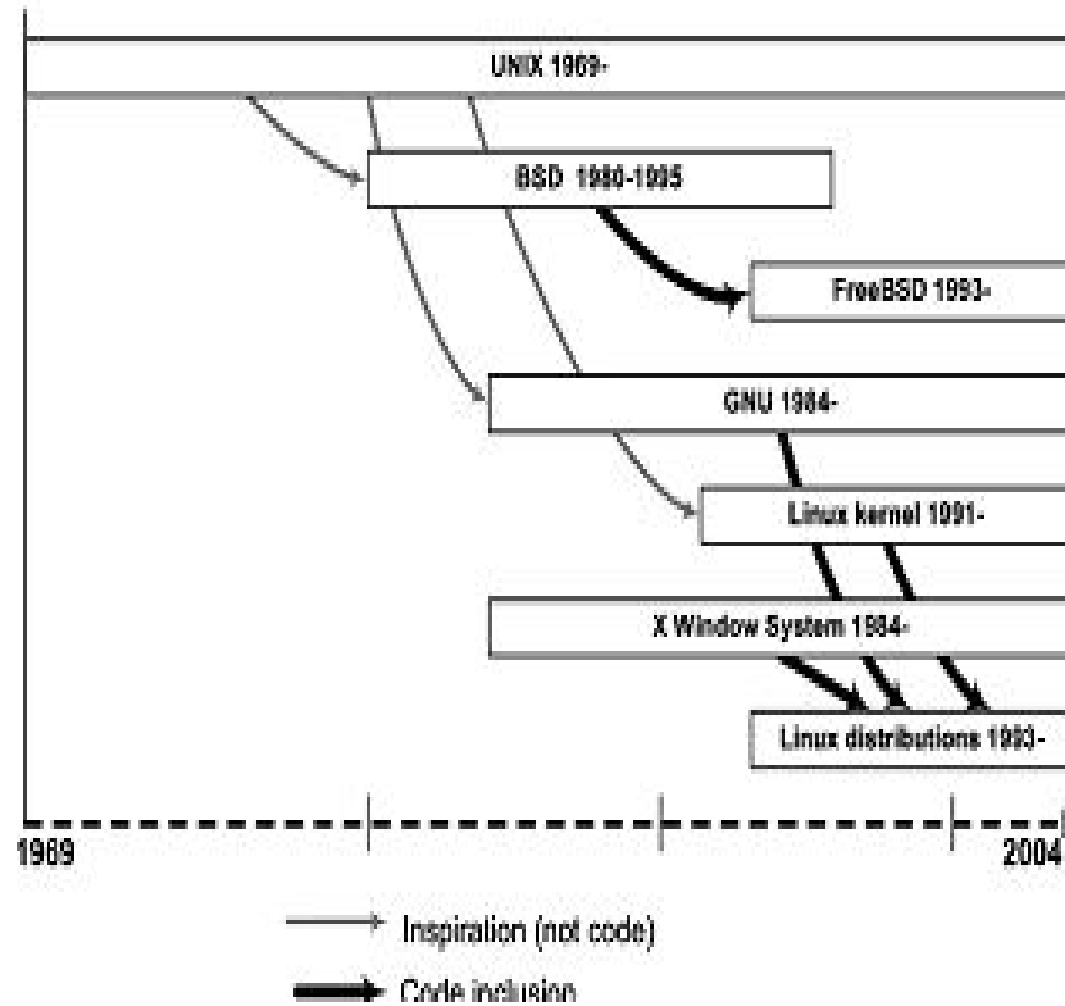
- Linux is a family of open source Unix-like operating systems based on the Linux kernel, an operating system kernel first released on September 17, 1991.
 - In 1991, a Helsinki University student named Linus Torvalds began developing a free UNIX kernel for Intel-based PC systems.
 - The Linux kernel was built with GNU Compiler Collection (GCC) and GNU utilities and relies on GCC for its portability. In early as 1992, by combining Linux with GNU, a complete operating system was constructed.
- 

Figure 1.1
Linux family tree.



The Personal Computer

➤ In the 1980s, the personal computer (PC) was introduced to business. Intel had introduced the first microprocessor in 1971, and personal computers such as the Apple II .

The Internet

➤ The first ARPANET paper was published at an ACM conference in 1967. By 1969, there were Internet nodes developed at UCLA and Stanford.

➤ By 1972, the file transfer protocol (FTP) and Internet mail addressing had been invented and demonstrated in Paris.



➤ From 1981 through the 1990s, the Internet, as measured by connected systems, users, and information transfer, was approximately doubling every year.

The World Wide Web

➤ Around 1991, HTML and the World Wide Web were invented at CERN, the European Center for Nuclear Research by Tim Berners-Lee.

➤ The World Wide Web, commonly known as the Web, is an information system where documents and other web resources are identified by Uniform Resource Locator

WHERE OPEN SOURCE IS SUCCESSFUL

1.3 Analytical Framework

- This section looks at how technology is adopted, when, and by which organizations.
- Important issues are disruptive innovation, lock-in, and the type of adoptions.

Disruptive Innovations


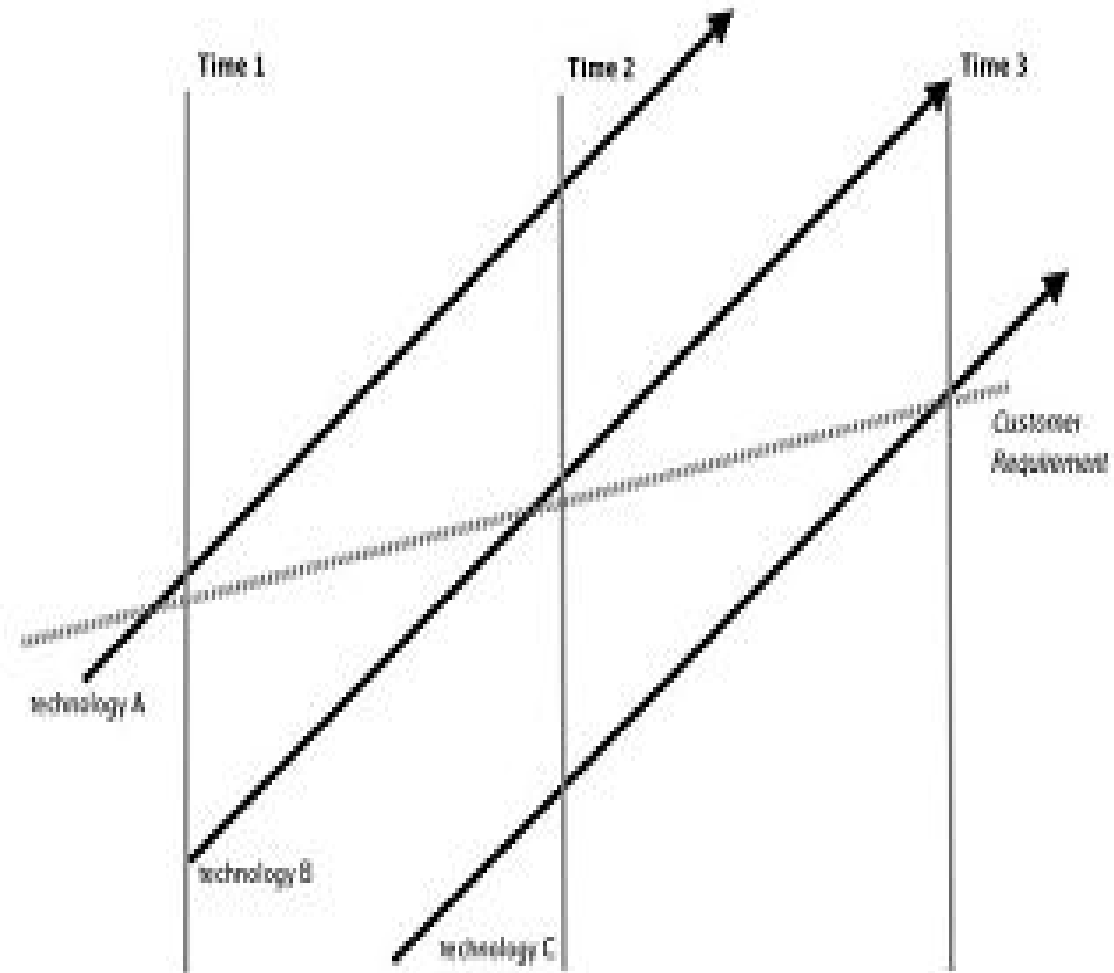
- A **disruptive innovation** is an **innovation** that creates a new market and value network and eventually disrupts an existing market and value network, displacing established market-leading firms and products.
- 

Figure 2.1
*Successive
disruptive
technologies.*



➤ We see three technologies, A, B, and C, whose performance (on the Y axis) is compared over time (on the X axis). A is always better in performance than B and B is better than C. A possible customer requirement is drawn with the dashed line.

➤ The customer need grows, but more slowly than the competing technologies improve. As a result, while initially only A will satisfy this customer, by Time 2 A or B will suffice, and by Time 3 C will also be sufficient for this customer.

➤ The two important assumptions in this model are:

➤ The new technologies are significantly less expensive or less difficult to use than the old.

➤ The switching costs must be fairly low.



Open Source as Disruptive Innovation

- Open source systems are usually less expensive than the alternative and often easier to use. In some cases, open source systems are a little more difficult to use.

Lock-In


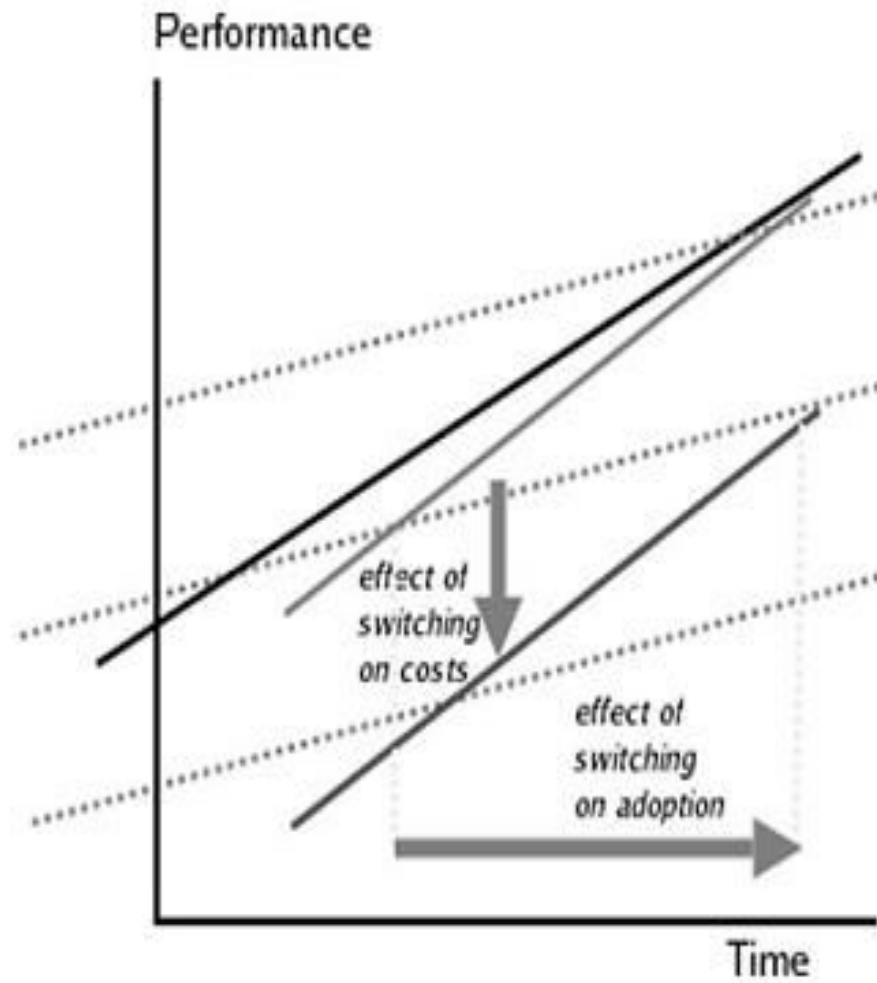
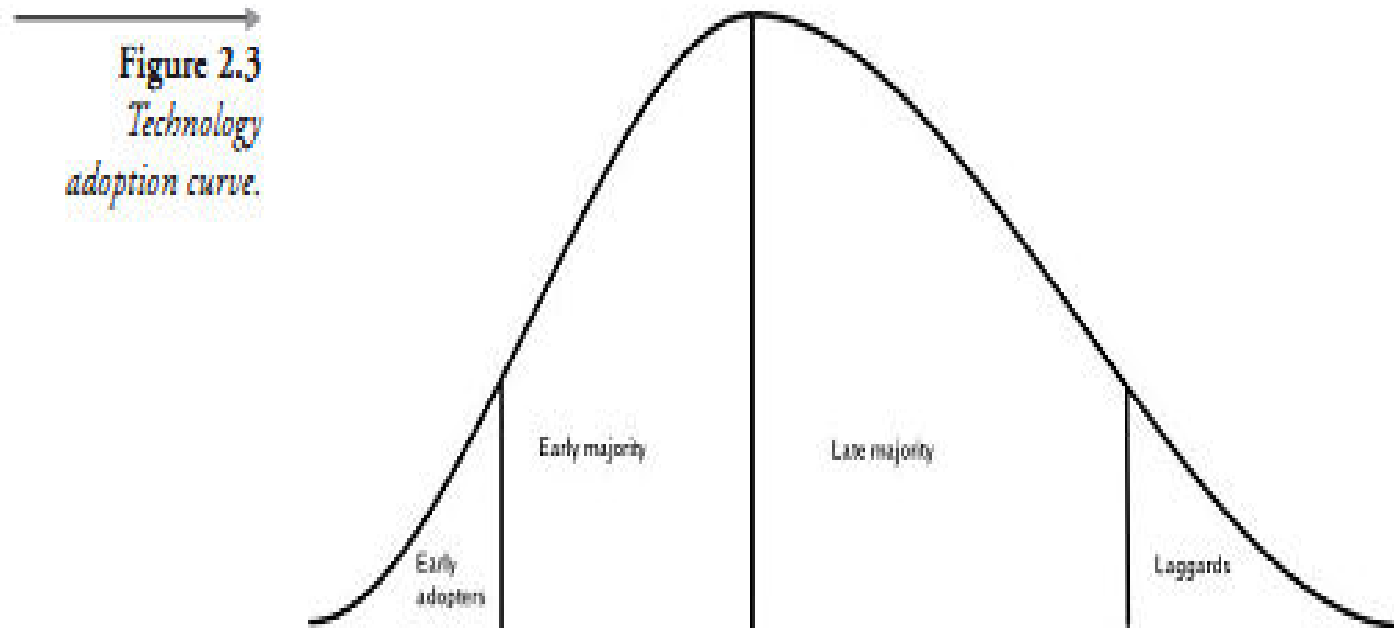
- From the customer's view, Figure 2.1 represents “commodification.” The customer has limited new needs and is ready to substitute less expensive products as they improve.
 - In software, with distribution costs close to zero, the product price will tend to zero and the only money to be made will be in service and training.
 - If switching costs are high, the new technology will have to be deployed into new uses until it is much better than the old, which could take years.
- 

Figure 2.2
*Effect of switching
costs on adoption.*



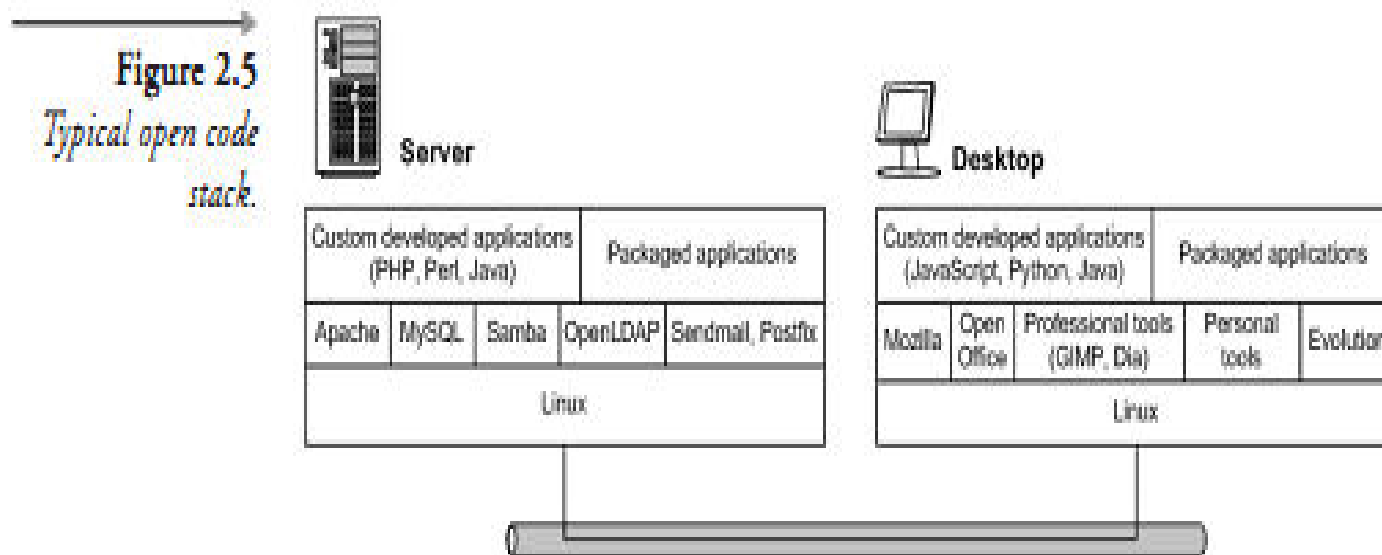
The Technology Adoption Curve

➤ More commented on the most difficult transition to make, which is the one from “early adopters” to “early majority.” Briefly, the early adopters are lovers of the technology and are often easily sold, accepting all kinds of problems and possibly buying without adequate cost justification for instance, for evaluations.



The Open Source Stack

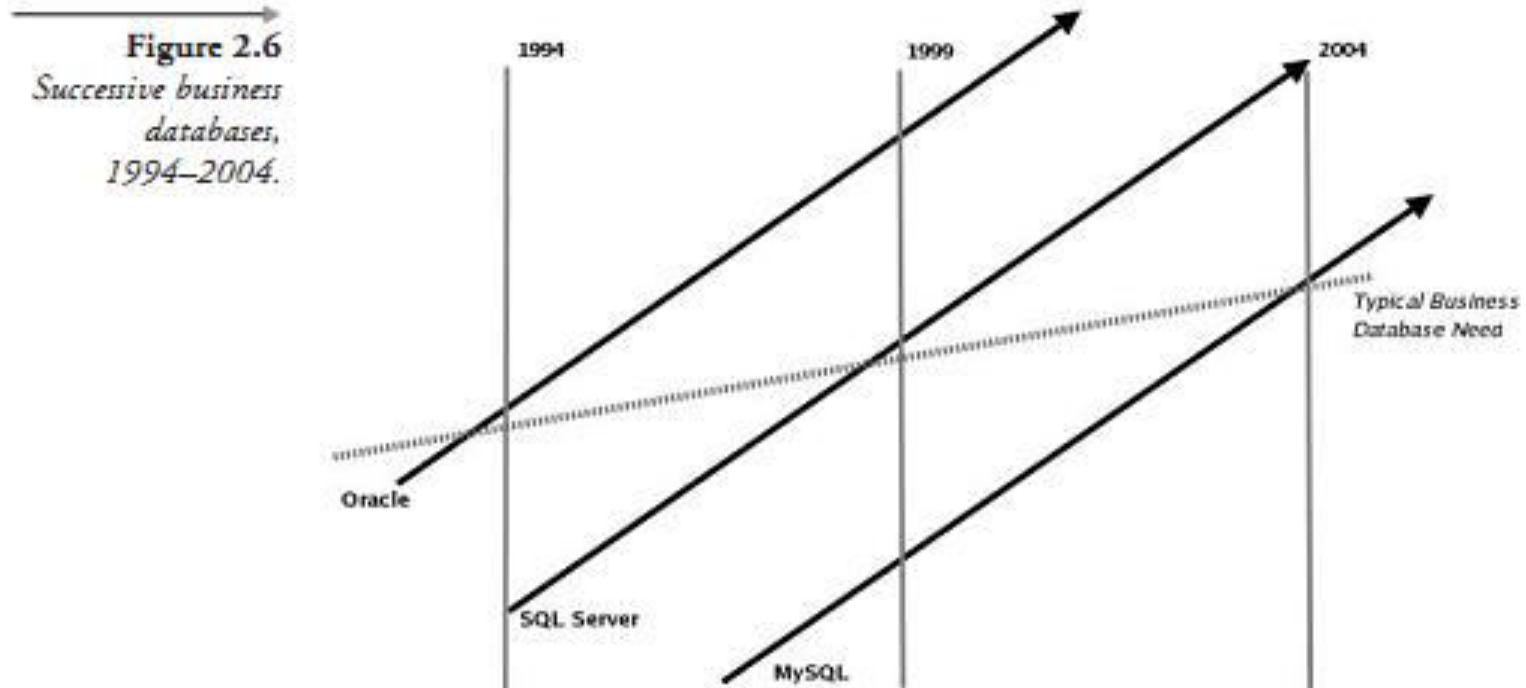
- It is possible to get open source products for all the elements of an organization's computer network. Most people will be using a mixture of open source and closed code, of course.
- Figure 2.5 is an abstract diagram that shows the layers of typical software on a server and a desktop system. Not all systems have all of these functions, of course, but most business systems will support these functions across the organization in a similar manner.



Adoption of Specific Open Source Technologies

➤ Adoption of specific open source technologies is:

- Database servers
- Web servers
- Operating systems
- Office suites



Database Servers

- Figure 2.6 shows the market for the database servers Oracle, SQL Server, and MySQL from 1994 through 2004.
- A **Database Server** is a computer in a LAN that is dedicated to **database** storage and retrieval.
- The **database server** holds the **Database** Management System (DBMS) and the **databases**. Upon requests from the client machines, it searches the **database** for selected records and passes them back over the network.


Web Servers

- A **web server** is server software, or hardware dedicated to running this software, that can satisfy client requests on the World Wide Web.

.



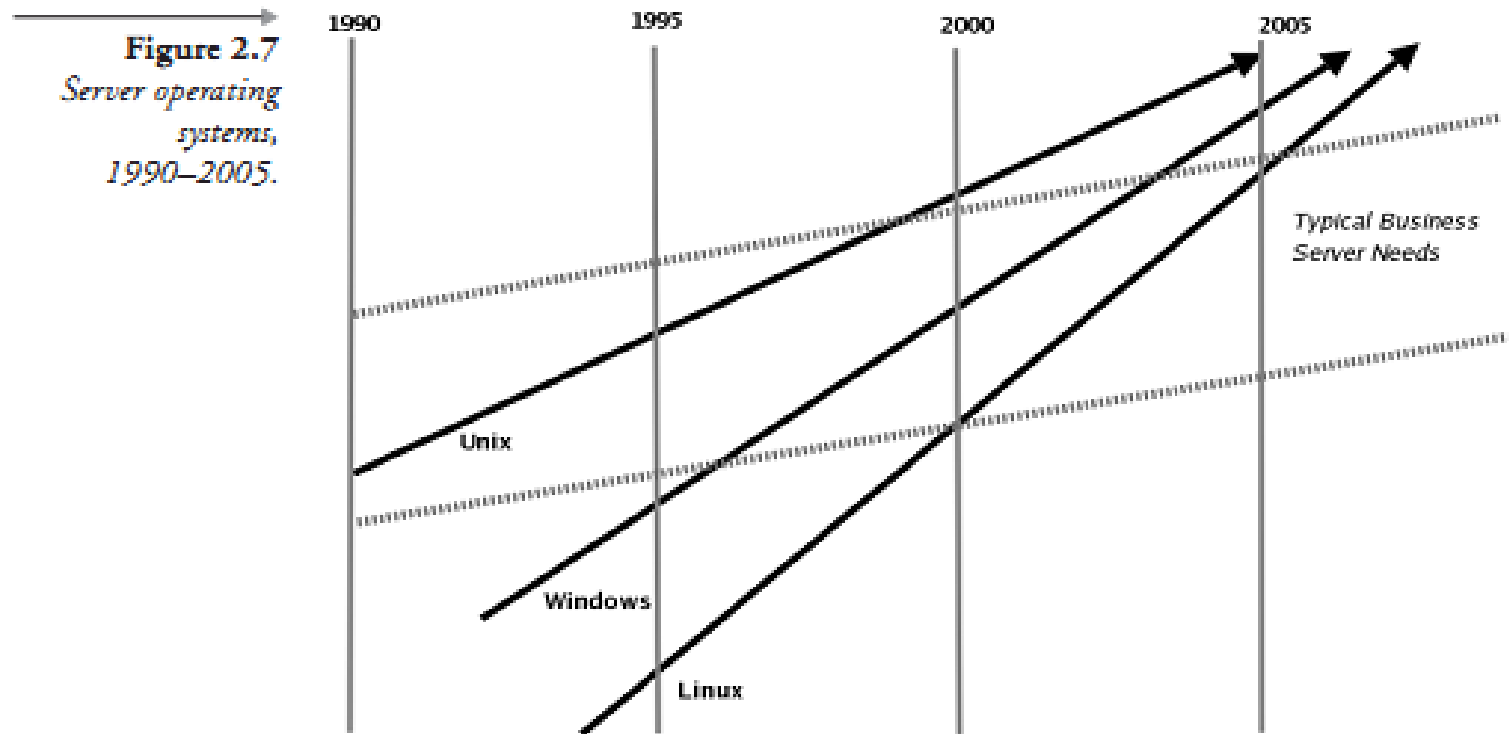
Operating Systems

- Linux was clearly a case of “good enough” but simpler and cheaper when introduced to commercial businesses around 1994.
 - Most early Linux customers were looking for a UNIX-like system that was inexpensive and ran on inexpensive Intel-based hardware.
- 

Office Suites

Open Office is a great example of a product that is “good enough.” It is not much different in functionality or look and feel from Microsoft Office.


If we have used neither before, we will most likely find them similar to learn and use, with more features than we need.



Queries ?

SHORT STORY

➤ **The Foolish Donkey**

- A salt seller used to carry the salt bag on his donkey to the market every day.
 - On the way they had to cross a stream. One day the donkey suddenly tumbled down the stream and the salt bag also fell into the water. The salt dissolved in the water and hence the bag became very light to carry. The donkey was happy.
 - Then the donkey started to play the same trick every day.
- 


➤ The salt seller came to understand the trick and decided to teach a lesson to it. The next day he loaded a cotton bag on the donkey.

➤ Again it played the same trick hoping that the cotton bag would be still become lighter.


➤ But the dampened cotton became very heavy to carry and the donkey suffered. It learnt a lesson. It didn't play the trick anymore after that day, and the seller was happy.

➤ **Moral of the story:**

➤ Luck won't favor always.



DAY 2

- ▶ **Previous topics:**
 - ▶ Open Source Software: Definitions and History
 - ▶ Definition of Terms
 - ▶ A Brief History of Software
 - ▶ **Where Open Source Is Successful:**
 - ▶ Analytical Framework
- 

Today's topics:

❖ Where Open Source Is Successful:

❖ Open Source Is in Widespread Successful Use

❖ Open Source: The Good, the Bad and the Ugly:

❖ What Is Good about Open Source

❖ Open Source Is Not Enough by Itself




1.4 Open Source Is in Widespread Successful Use


The Linux and FreeBSD operating systems, the GNU compiler suite, and the Apache Web server are massive software systems used all over the world.

They are large and sophisticated and complete with the largest closed code systems, such as Windows.

Open Source Is the Heart of the Internet

There would be no Internet without open source. Open source software has been in use on critical parts of the Internet, including mail routing, domain name assignment, and key components of the TCP/IP network stack.




- ❖ TCP/IP
 - ❖ Send mail was written as open source in 1975
 - ❖ BIND, the program that maps server names to IP addresses
 - ❖ Apache is the most used Web server
 - ❖ Linux is the operating system
 - ❖ Mozilla, the browser
- 

Linux Is Shipping a Lot

- Linux is the shipping server worldwide (after Windows).
- The server growth rate was 35 percent for Linux and 14 percent for Windows
- There will be approximately 25 million Linux systems in use

Open Source Appliances Are Everywhere

- Appliances based on Linux and open source tools have been widely deployed in corporate networks, small businesses, and homes for several years with proven reliability.
- 

➤ Examples of these consumer, home office, and small business devices include the following, all based on Linux:

1. TiVo television recorders
2. Mirra network-attached disk drives
3. Cobalt (Sun) Web and mail server appliances
4. Pogo database (MySQL) appliances

New Companies and New Businesses Use Open Source

- When a new technology becomes available, there are many obstacles to its rapid deployment.
- First, if we have done the intellectual and physical work to build a satisfactory networking solution for our organization based on Windows 2000, then a better or cheaper solution, whether based on Windows 2003 or Linux.



➤Second, any savings are probably negated by our sunk costs, at least until some major upgrade comes along.

Dot-Coms


➤A number of powerful companies have come from the dot-coms Internet industry.

Bioinformatics

➤Bioinformatics is a rapidly growing industry characterized by scientists using powerful computers to perform gene sequencing, cladistics, computer-driven ecology, modeling of dynamic systems, and other computer-intensive processing.



Open Source Is Broadly Adopted

- Most top computer companies are strong advocates of open source software for significant elements of their products.
 - This includes all of the top integrated systems companies, such as IBM, Hewlett-Packard, Apple, and Sun. IBM and HP make billions from Linux sales. Sun has several product lines based on Linux.
 - The Apple Macintosh OS X operating system is based on the open source FreeBSD operating system.
- 


Other large corporate adopters of open systems include:

- Ford
 - L.L. Bean
 - Delta Airlines
 - Merrill Lynch
 - Telstra
 - Cisco
 - UBS Warburg
- 
- A decorative graphic element in the bottom-left corner of the slide, consisting of overlapping blue and black geometric shapes that form a triangular, abstract design.

Open Source: The Good, the Bad, and the Ugly

What Is Good about Open Source

Advantages of using open source software include our ability to:

- View source code
 - Change and redistribute source code
 - Buy from different vendors and adopt new platforms
 - Avoid proprietary information formats
 - Allow integration between products
 - Reduce software licensing cost and effort
 - Develop and deploy effectively internationally
 - Draw from a large pool of skilled professionals
- 

Why Your Right to View Code Matters

There are reasons why you should have the ability to review source code, even though you don't expect to do it, don't want to do it, and may not even be able to (if you don't know C++ or Perl).

If there is a problem with a system you are maintaining, you may not read the code but you will do the following:

- Check the documentation that describes how the system should work, including FAQs, to see if you are using the system correctly.




- Check support base or bug reports for the system for similar problems and solutions.
- Use search engines such as Google to search for similar problems and solutions.

These methods are all improved by source code availability.



Why Your Right to Change and Redistribute Code Matters

There are reasons why you want to be able not only to view the source code but to change it and possibly redistribute it.

- First, code has to be kept up-to-date to be worth seeing, and that mechanism must be public.
 - Second, as a practical matter, we have found that the ability for serious users to contribute is a real source of strength for the large open source projects such as Linux and Apache. It has allowed them to improve more rapidly than the largest and best closed code competitors.
- 

➤ **You Can Buy from Different Vendors and Adopt New Platforms**

➤ You do have to plan and build for a capability to buy from multiple vendors; if you do, your investment in software and business practices is safe than if you rely on one closed code platform. This has been proven repeatedly in the past.

➤ **Open Source Avoids Proprietary Information Formats**

➤ It is unusual for open source to implement proprietary data formats, but it is not impossible. Open code could use proprietary keys to make data inaccessible even though the algorithms are public, as is done routinely for security reasons.



➤ **Open Source Allows Integration between Products**

➤ With open source, because data and code specs and source code are available, we can always make things work together by examining these documents. With proprietary systems, we are dependent on the code owner to make changes (or not).

➤ Proprietary systems are more likely than open source to offer canned and integrated “solutions.” Unfortunately, these are rarely more than marketing fluff.



➤ **Open Source Licensing Is Simpler and Less Expensive**

➤ **Open Source Is Less Expensive**

➤ Open source saves money over closed code software in almost all cases. The logic is simple.

➤ The software costs much less, while all other costs can be expected to be approximately the same or less unless some special circumstances apply.

➤ I'll use Linux in the following examples, but the same would apply to other products, such as My SQL.

The real advantage of open source in licensing costs is swamped by some other large numbers. This can be done in a variety of ways:



- Use expensive hardware (e.g., putting Linux on a mainframe).
- Use expensive development, management tools, or services in the Linux .
- Use large numbers for development, training, support, or downtime; then estimate Linux as a little (or a lot) worse than the closed code solution in this area.

Open Source is a Good Solution for International Companies

Everyone in the world is not going to license closed code software at today's prices. Today, that has led to heavy use of unlicensed software in many countries.

If an organization is operating overseas, the cost of software may already be an issue.

The decision to make software available in a particular language and culture is a difficult one for a closed code company.

There Is a Large Pool of Skilled Open Source Professionals

At the present time, with the majority of servers and an overwhelming majority of desktops running Windows, there is an approximate parity in cost and availability of skills between Linux and Windows.


Some surveys or studies find one or the other is easier to staff, and there are substantial regional variations.




Open Source Is Not Enough by Itself

It may be a good thing that a system is open source, but it is not sufficient. It is commonly argued that open source generally has certain good features by virtue of its construction for example, that it is secure, virus free, reliable, or at least more so than closed code software.

We will want to review any open source product for openness of each component and standard employed. Once we have an open source product we want to use, we will want to review at least the following elements:

- Platform portability (hardware, OS)
 - Database portability (stored procedures, APIs)
 - Language
 - Data architecture
 - Software quality (modular, documented)
- 

Deployment Platform

- We would prefer to see applications that can be deployed on standard, scalable computer hardware available from several different vendors and employing internal standard components obtainable in an open market process.
 - We would prefer applications that can support a range of operating systems, since this will provide the most choices.
 - Ideally, this would include at least Linux, Windows, and other UNIX. Alternatively, we would prefer applications that run on an operating system available on a range of hard-ware and from multiple sources in practice, this is Linux.
- 


Database Platform

If the application uses a database, we would like it at a minimum to use a standards-based (SQL) database and a consistent, well-designed current interface to that database's APIs and stored procedures. It would be preferable to have a choice of databases, particularly because some databases require particular platforms for the underlying hardware and OS.

This is not always going to be possible, since the extra work of supporting multiple databases is not always justice for an application.

Software Language, Architecture, and Implementation

We want the application to be written in an open source language, preferably one that is portable between platforms. This set of languages generally includes C/C++, Perl, Python, PHP, and UNIX shell. Java is not an open source language.




The Java specification is not open, although there are competing implementations of the language that are open source software. However, it is cross-platform, and much open source software is written in Java.

It is particularly important to review the software architecture to identify dependencies on code libraries, platform APIs, and so on. Most languages are portable until those issues are considered.

Data Architecture


The important issues with application data architecture are that it allows external access to all data through well-understood standards. For example, SQL or XML.

In some cases, applications may impose per customer fees or license-based restrictions on access to data. So in these cases, you know where your data is, you have figured out how to get it and display it, and you still owe a fee or have to ask permission to use it.



QUERIES ?

DAY 3

- ▶ **PREVIOUS TOPICS :**
 - ▶ **Where Open Source Is Successful:**
 - ▶ Open Source Is in Widespread Successful Use
 - ▶ **Open Source: The Good, the Bad and the Ugly:**
 - ▶ What Is Good about Open Source
 - ▶ Open Source Is Not Enough by Itself
- 

Today's Topics

- ▶ How Choosing Open Source Is More Difficult for You
- ▶ What Others Say about Open Source.




How Choosing Open Source Is More Difficult for You

1. Open Source Has a Less Complete Level of Sales Support

No Direct Sales Model


There are not many products, particularly in the early stages of adoption that sell themselves. Without a direct sales force, many products are unlikely to be adopted.

Indeed, otherwise good products may not even be considered. Microsoft Test, for example, was a product as good as the competition, but it never sold since it was too inexpensive and had no direct sales force; eventually it was given away to Rational.



The sales support that a large vendor can offer (for or against) can include executive-level pressures. In some organizations, it is difficult to counter these from technical positions.

A good vendor:

- Puts on sales demonstrations, provides references, and prepares information on costs and competitor
 - Meets or bypasses objections from IT traditionalists
 - Helps manage senior management buy-in
 - Provides technical demonstrations and may help set up a tech lab
 - Offers installation support and advice to evaluators
 - May offer consulting or other help to make the first case successful
- 


2. Specific Product Reviews Will Not Favor Open Source

Product reviews are generally conducted fairly and in a reproducible manner by an independent company and in that sense are fair.


There is no shortage of general information from reviewers and vendors indicating that open source in general and Linux in particular are important upcoming trends.

This book has already quoted several of these statements, and they continue to be published regularly.


The views of companies such as Gartner and IDC have been very helpful in spreading the word in these areas to organizations that might otherwise have been uncertain or unnecessarily conservative.




To look at this in more detail, there are four ways that specific product reviews can be misleading. The controlling party selects:

- The package of products under consideration
 - The timing of the review against release dates
 - The competitors to compare against and the versions of the product to compare
 - The behavior being evaluated (e.g., performance, stability, ease of use)
- 

Package of Products

- The differences in the way open source and closed systems are packaged and sold can lead to misleading comparisons.
 - Open source distributions generally include office suites, various personal tools, databases and other servers, and development tools, often several of each kind.
 - These are generally automatically installed for you if you select a profile such as “professional” or “developer,” and at no additional charge.
- 

Timing Reviews against Release Dates


- There are differences in the timing of releases and the reviews. This one ought to balance out but it systematically favors the closed code vendors, because they fund the reviews and thus pick the timing.
 - So the review will compare a late beta product with its new features against an older product from open source.
- 

Competitors and Product Versions

Another issue is product versioning. The same review which compared Red Hat Enterprise Linux 2.1 (RHEL) with Microsoft Small Business Server did not compare similar products.

Small Business Server is a low-end version of Windows Server with restrictions on number of users and through-put, bundled with a limited-use version of the SQL Server database and some other products, such as a fax server and a wizard-based front end, which were included specifically to improve ease of installation for small businesses.

RHEL is the version of Red Hat Linux designed and priced for use by large (“Enterprise”) businesses, having, for instance, clustering and a choice of full-function database packages. RHEL should be compared with the more expensive Microsoft Windows Advanced Server.



Evaluation Criteria

Criteria being evaluated are selected by the reviewer to favor the comparison. This includes what is measured and which products are selected for competition.

For example, Microsoft prefers to evaluate application development as .Net versus Java, because Java tools are more expensive and harder to develop with than, say, PHP.




3. Open Source Products Are Not Bundled, Branded, or Integrated

Bundling is when several products come packaged together. There is generally some, often limited, integration.


Branding is when several products are named similarly so that all can benefit from the reputation of some.

Bundling

Closed source products have several incentives to pursue bundled software (that is, to put together several solutions with incentives to buy them together), branding strategies (associate products with each other), or to create direct tie-ins between products.



Branding

- Microsoft has historically sold three different operating systems under the Windows brand: Windows 98/ME (now obsolete), Windows NT/2000/ XP, and Windows CE.
 - This is a branding strategy.
 - There are some common characteristics, such as a similar look, and an attempt to make tools such as SQL and VB run across the platforms, but these are in fact three totally different code bases.
- 


Integration

The level of integration offered by closed code platforms is often disappointing, particularly if you assume it.

In particular, if what you want is not available you may have no way to proceed at all.

Closed code platforms often offer the integration they do because they are closed; you need integration and you can't do it without access to the code, but they won't let you access it, so they have to do it for you.

And, of course, integration between closed code products from two vendors is very difficult. It is best to approach any two products with a needs analysis and a review of what is available.



What Others Say about Open Source

These are some questions we hear repeatedly, often asked by vendors who compete with open source software.

Those questions include whether open source is financially viable, if it will fall apart or fade away, and if it is really less expensive.

Can Vendors Make Money from Open Source?

This question is most often asked by vendors that are competing with open source products.

Customers are usually untroubled by lower prices in products they buy, if arrived at legally.

It turns out that many vendors are making significant revenues from Linux today.




What If Linux “Fractures” Like UNIX?

The specific reasons for the fracturing of UNIX are complicated and do not appear to be present with Linux.

One problem with UNIX was the trademark issue.

Linux can be freely used without trademark acknowledgment to another company so there is not the name fragmentation around Linux that occurred with UNIX, which was marketed variously as Solaris, HP-UX, Xenix, Irix, AIX, Tru64, and SCO Xenix, among others.



Is Open Source Really Cheaper When You Consider All the Costs?

Vendors may point out that if you price a system including middleware, such as IBM Web Sphere or BEA Web Logic; database software, such as Oracle; or hardware, such as a mainframe, a solution involving open source may not always be cheaper (or much cheaper) than a system with-out open source.

Maybe it is not always cheaper in those cases. Customers don't always want cheaper.

Maybe they like the solutions involving Web Sphere, Web Logic, Oracle, or Linux on a mainframe and think they are worth the money.



QUERIES?

UNIT - II

Five Immediate Open Source Opportunities: Create an Open Source Lab- Migrate Infrastructure to Samba and Open LDAP- Build Some LAMP Applications- Bring New Desktop Systems to the Underserved- Migrate Applications and Databases to Open Source- Five More Open Source Opportunities: Introduction- Directory Services- Email- Groupware and Collaboration- Complex Web Publishing- Manage User Desktops- Other Possibilities.

DAY 4

- **Today's Topics**
- Five Immediate Open Source Opportunities:
- Create an Open Source Lab
- Migrate Infrastructure to Samba and Open LDAP
- Build Some LAMP Applications

Five Immediate Open Source Opportunities

2.1 Create an Open Source Lab

The first step to take is to set up an open source lab. This supports all the activities that follow, in this book and in the organization, so it is a critical beginning.

From experience with other migrations, such as in the personal computer and network eras, we know that using a lab is a pivotal move.

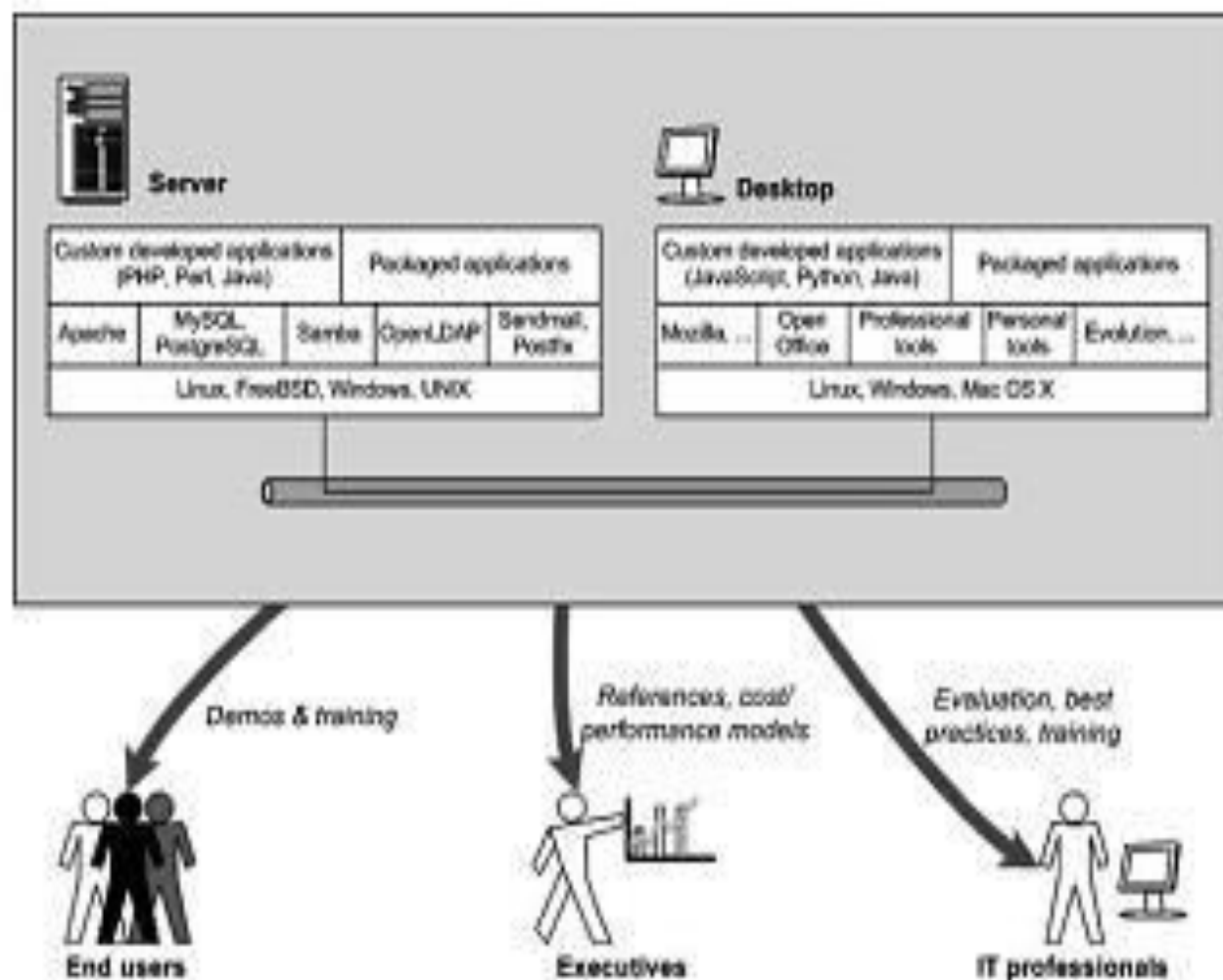
It is a first step to take for an organization that is considering open source.

The lab is really a communication tool to discover useful information, activity, and resources and bring it to the attention of decision makers and professionals who can act on it.

Figure 4.1 shows the various open source elements on display that are used for communicating with key decision makers, including end users, IT professionals, and executives.

It can be a good place to conduct training in relevant open source skills for developers, administrators, users, and executives.

Figure 4.1
Open source lab.



Review Existing Work

Review existing work to see where open source is already in use in the organization.

Look for existing use of open source products in your own organization, such as:

- Outsourced services such as Web sites or Web services that may use open source
- Software packages that have been installed on open source or are under consideration and could be so installed
- Departments or divisions that have made or are considering unilateral moves
- Individual evangelists or developers who are considering or using open source

Departments or individuals who have already deployed open source, the likeliest candidates are:

- Small Web sites using Apache and PHP
- Small databases using MySQL
- Scripting using Perl or UNIX shell scripts on Windows
- Use of Samba for Windows file sharing
- Individual Linux desktops
- Appliance servers for Web, caching, network, storage, and firewall / security

Train Developers to Program in Open Source Languages

We can train Microsoft Visual BASIC developers to develop in the open source programming languages PHP or Python.

These languages are easy to learn and productive to use, and there is no charge for their installation or use by developers or end users.

They can be learned and used on Windows, since they run as well on Windows as any other language.

2 Migrate Infrastructure to Samba and Open LDAP

1. File and Print Server

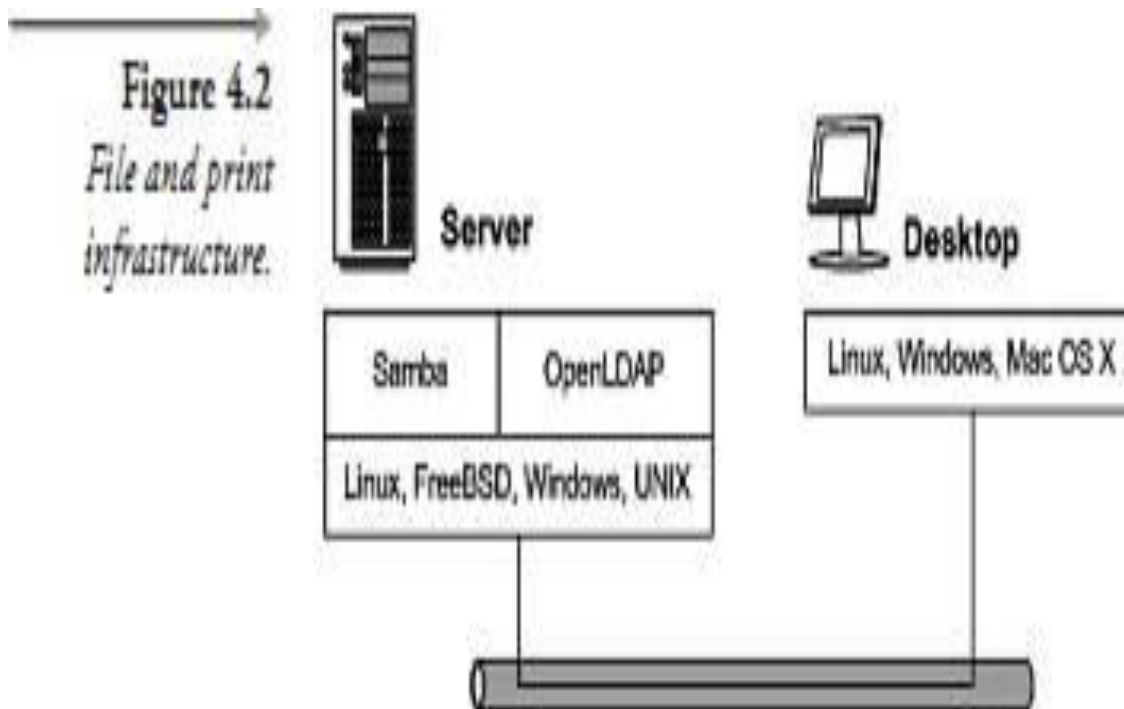
According to the Gartner report “Fear the Penguin” (January 2003), by 2004 most organizations will, as a standard practice, be deploying file and print servers based on Linux.

Essentially, this means using Samba for file and print sharing.

Linux servers can use Samba combined with print management such as CUPS to serve Windows clients without change to the clients.

Samba clients (and servers) ship with current Linux and Mac systems.

It could be any UNIX system, including Mac OS X or FreeBSD, or it could be network-attached storage appliances. But Linux is probably going to offer the best price/performance.



Hypothetical Analysis

Hypotheticals are possible situations, statements or questions about something imaginary rather than something real.

The current Linux run rate is higher than UNIX and Novell combined, and is forecast to match Windows during 2004.

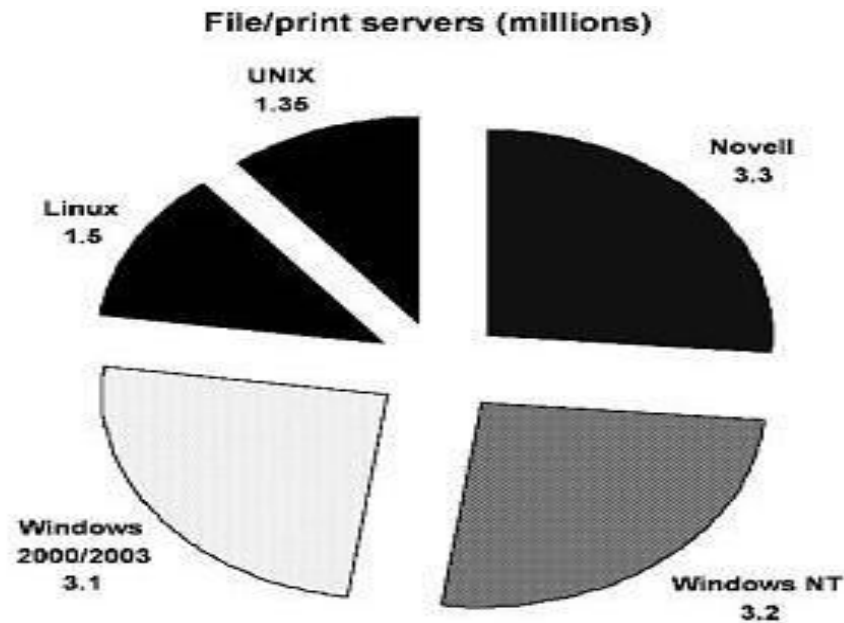
As UNIX systems used for file and print age, the natural migration path for these systems is Linux.

 **Table 4.1** *Installed Server Systems 2003*

Server OS	# Installed (millions)	# File/Print (millions)
Windows	9.5	6.3
Novell	3.3	3.3
UNIX	3	1.35
Linux	2.3	N/A

The Windows systems, over 3 million servers, are simple Windows NT4 systems serving file and print services with NT4 authentication, which provides easy migration targets to Linux and Samba.

Figure 4.3
File/print installed systems.



In replacing infrastructure servers, we can choose to:

- Continue to use a distributed design but use Linux servers
- Consolidate onto fewer larger platforms.
- Replace servers with network-attached storage.
- Combine all three strategies.

Many Small Linux Boxes

The problems of managing many small computer servers for file and print serving are well understood, which is what generates interest in the alternatives.

Consolidation

Server consolidation is a way to make a more efficient IT environment by combining **servers** or replacing legacy **servers** with virtual systems.

Server consolidation is very likely to be a piece of the solution, because it puts configuration and updates under better control.

The arithmetic of consolidation can be attractive, but unfortunately organizational boundaries and geography often prevent us from getting the full benefits because some small servers just need to be in remote locations or outlying divisions.

Network-Attached Storage

Network-attached storage (NAS) offers management and cost advantages, although once again actual geography and the need to replace hardware that is a sunk cost can eat up the advantages.

Manage Use of Windows Proprietary Features

The goal here is for those organizations that are not able to make a large-scale move from Windows today to prepare the ground for a move away from Windows to open source when the time is right.

This could be in response to a business change, license upgrade, or just when systems are ready for replacement.

Train Administrators in Linux and Samba

We should train MCSEs in Linux, including Samba and CUPS (file and print sharing), and, if appropriate, Open LDAP and mail administration.

There are about a half million MCSEs and 1.5M MCPs.

These people are often advocates for the technology skills they have now, particularly since it may have taken significant time and money to acquire them.

It will be helpful to teach them the open source equivalents

Build Some LAMP Applications

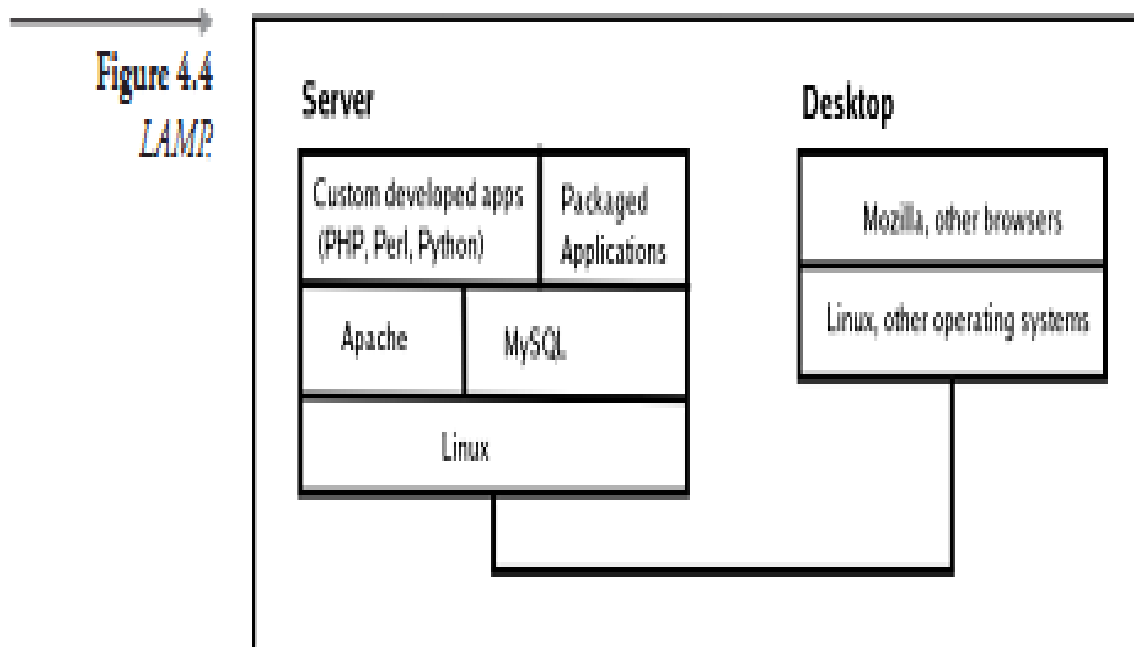
An excellent first step is to build one or two simple applications using Linux and Apache with MySQL or PostgreSQL as a database and PHP, Perl, or Python as a development language.

This combination, known as LAMP, allows you to build a pure open source system quickly.

You are then free to compare the ease of construction and maintenance and the running costs to alternatives that your organization has used.

MySQL is a powerful database that is easy to install and learn to use, and the LAMP combination makes a nice clean story in terms of licensing and costs, particularly for distribution outside your organization.

An alternative open source database is PostgreSQL.



Web Applications

For most people, these will be Web applications using PHP with Apache.

They could equally well be external-facing sites, such as Web presence or ecommerce; business to business; or internal-facing sites, such as an employee portal.

Client/Server Applications

➤ Client/server applications can be built also, if that's what you need. Everybody builds Web applications these days, but there is still a type of application that can be less suitable to the Web and more appropriate for client/ server.

- Authors
- Call center
- Customer relationship management
- Decision support or spreadsheets
- Engineering workstations and CAD
- Analysis (financial, mathematical, economic, etc.)
- Medical

Selecting the First System

At this time, you are not likely to be building the first Web application for a company of any size. Likely opportunities to try the new approach may include:

- Building a new Web site or application
- Adding significantly new functionality to an existing site or application
- Migrating an application developed with obsolete tools
- Migrating Web development, such as ASP to ASP.Net or JSP
- Development or management of the site is being moved, such as out-sourced or brought back in house

Queries

DAY 5

PREVIOUS TOPICS :

Five Immediate Open Source Opportunities:

Create an Open Source Lab

Migrate Infrastructure to Samba and Open LDAP

Build Some LAMP Applications

Today's Topics to Discuss:

Bring New Desktop Systems to the Underserved

Migrate Applications and Databases to Open Source

Bring New Desktop Systems to the Underserved

Simply make open source software available to users who currently don't have and cannot afford a current licensed set of desktop software, including Office, Visio, Photoshop, SQL Server, and Visual Studio. There are no migration costs. The systems will exceed expectations.

Users who have desktop software today include many in these categories:

- New and small businesses
- Franchised businesses or dealerships, and other business partners
- Customers and others accessing our Web site
- Our home, family, and neighbors
- Community organizations, such as schools and healthcare
- Retail, distribution, and manufacturing workers

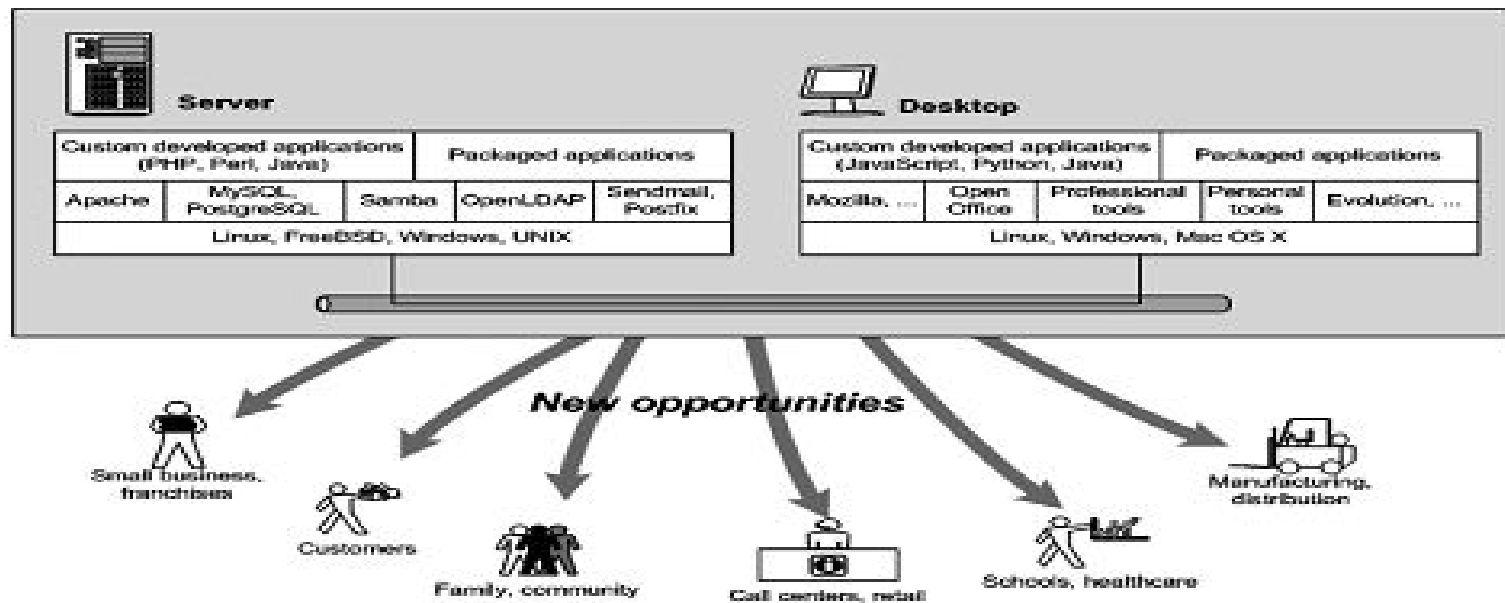


Figure 4.5 *New systems.*

- Call centers
- Manufacturing and distribution
- Retail and hospitality
- Government
- Education
- Healthcare

New and Small Businesses

New, small, and home office businesses have the opportunity to set up without migration costs and without the difficulty of communication change to a large workforce.

Franchises

With many retail operations, most of the stores are not wholly owned subsidiaries but instead are franchised. In fast food and hospitality, there are often regional large franchisees that operate dozens or hundreds of stores for one or several franchisers.

In the franchise situation, it can be difficult to specify particular systems consistently, since the organization buying the systems is not the one specifying. This can lead to several problems:

- Similar systems in different stores implemented with different platforms, so they don't work with central systems that must support them
- Different systems in a store implemented with different platforms, so they don't work together in the store
- Lack of ability to get volume discounts or other leverage, because hundreds or thousands of purchases are split into many small purchases across many organizations

2.4.3 Call Centers

Approximately 2 percent of the U.S. workforce works in call center or data entry functions. Call centers generally have hundreds of workers on a few dedicated applications. They have these issues:

- Applications are sometimes on legacy boxes
- Need to keep cost low
- Management of workstations should be centralized and simplified, with no unnecessary extra applications or customization

Retail, Food Service, and Hospitality

Retail includes companies such as Office Depot, PetsMart, and Wal-Mart, which operate many hundreds or thousands of stores, as well as individual stores and smaller chains, which may have one or a few sites.

Automobile dealers are a large retail category in themselves, and one that has often not had very effective computer systems.

Food service includes less expensive eat-in restaurant chains, such as Red Lobster, and fast-food companies, such as Burger King, as well as high-end restaurants.

The hospitality sector includes hotels, casinos, cruise lines, and other places where people stay. I will call the location of the point-of-sale system (POS) in all of these cases a “store” in the following text, although it might be a restaurant, hotel, or other location.

Government, Healthcare, and Education

Government, healthcare, and educational systems all share similar problems across geographies and have no good reason not to cooperate.

Government

It is not common to think of governments as technology leaders, particularly in modern American discourse, but government is often the only vehicle for funding early technologies.

Much of the early development of computers, PCs, and the Internet was government funded, often related to aerospace (e.g., NASA) or defense (e.g., DARPA).

Government is a very good fit in some ways to open source at later stages. Government customers outside of favoured programs in defense and aerospace are often acutely concerned about cost.

Governments deal with issues of fairness.

Healthcare and Education

In many countries, healthcare and education are wholly or partially part of the government. In any case, many of the issues, such as very large scale and irregular and inadequate funding are the same.

Cost is the largest issue in healthcare; it has been growing at such a high rate over the last 20 or 30 years that early attempts at managing the problem have stalled.

Healthcare also has a very particular problem of allowing many parties cooperating around an individual set of records, while managing privacy.

The various parties do not have a high level of trust in each other.

Education systems typically have very large numbers.

Giving a mailbox to every child in a school district, for example, can lead to very large volumes of mail traffic far above corporate levels.

There is a bug training and ease-of use problem, since training is often not commensurate with the systems deployed

Unlicensed Software

Open Source Is a Legitimate Alternative to Piracy

An important case for open source is organizations that have not currently purchased a satisfactory system based on Windows.

There is a large group of organizations that never licensed the software they are using. According to the industry group Business Software Alliance(BSA), 40 percent of the world's software is not legitimately licensed.

Piracy Touches Most People

While software piracy is unusual in larger U.S. organizations, and many professionals may not have experienced it there, it is common in smaller businesses, in situations such as franchise operations, among students, and in the home in general.

I think we have all seen this going on at a small business or a neighbour's house.

There are large flea markets outside every U.S. city every summer weekend selling this stuff.

International Opportunities

Consider the new systems that will be deployed internationally over the next few years. The computer industry continues to develop systems faster than the old ones go obsolete, and the price continues to drop.

Thinking More Broadly

We can take the idea of the underserved much further. We know that when a new technology is introduced, we tend to use it initially to replace previous technologies in similar uses. Only later are breakthrough uses conceived.

Migrate Applications and Databases to Open Source

There are many applications and databases on older and smaller servers that have been rendered obsolete by the Web. We discussed the statistics of installed servers previously.

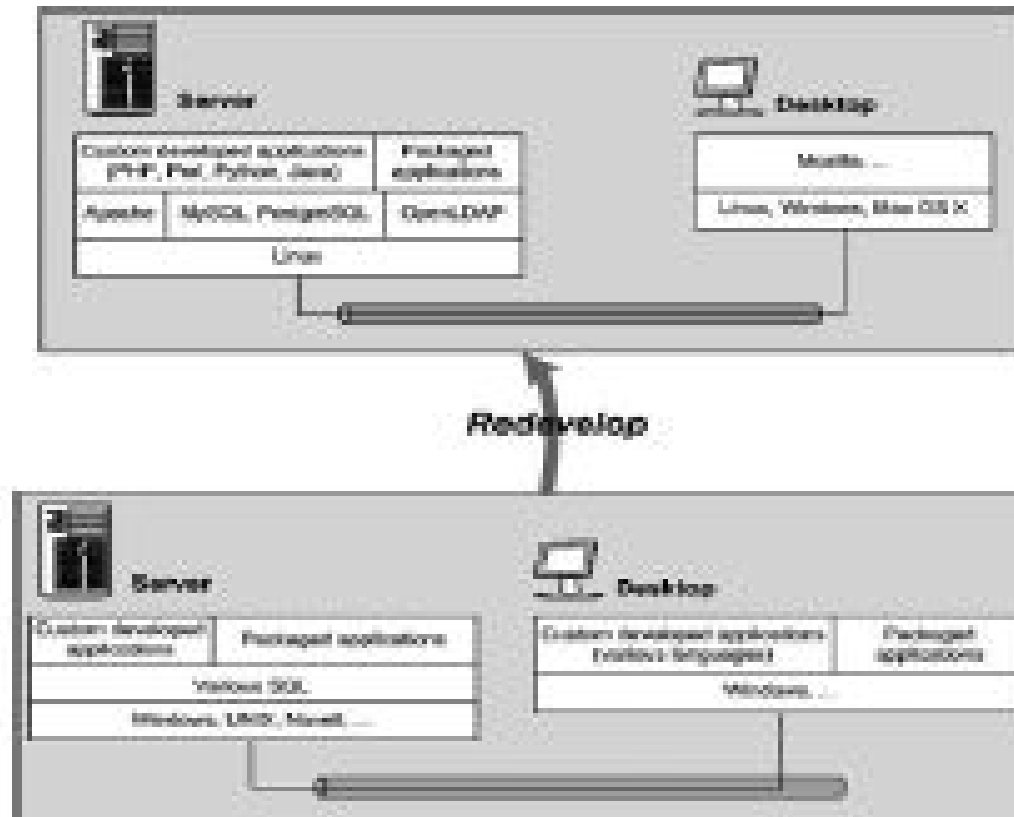
The same analysis of the server installed base reveals that 25 percent of Windows servers and 40 percent of UNIX servers are application and database servers.

This is approximately 2.4M Windows servers and 1.2M UNIX servers.

There are also a smaller number of mainframe servers with a lot of big applications and databases on them that will need to get moved to new technology at some point.

Most of these applications were written before the Web with databases and tools that are now obsolete.

Figure 4.6
LAMP
redeveloped.



If a system is just old and a sunk cost, it is not likely to be migrated until something happens to make it suddenly expensive or needing to change. Specific reasons to migrate include:

- The old system is obsolete and the manufacturer is cutting off support.
- The old system needs an expensive upgrade by some date or will cease to work.
- The company is merging, splitting, or downsizing and would pay disproportionate costs to stay on system.
- The system is essential but particularly difficult to maintain or ineffective in supporting customer needs.

Evaluate Open Source Databases

While most companies will continue to use a nonopen source database, usually one or more of DB2, Oracle, and SQL Server, there are situations for almost every organization where an open source database will be adequate to the task, less expensive, have a smaller installed footprint, and be easier to manage.

Some organizations may have the opportunity to move entirely to open source databases.

Replace Small and Old Database Applications with Open Source

Because open source databases are not expensive, they can replace the category of systems that could not be built with Oracle or SQL Server for price reasons in the past, but were built with Access, Paradox, Progress, Borland Interbase, xBase databases (such as FoxPro), and other personal databases.

Migrate UNIX to Linux

Another migration to consider is the UNIX workstation. Millions of these were installed from the 1980s on for engineering workstations, including mechanical CAD, software development, and so on. Manufacturers included Apollo, Sun, HP, IBM, Silicon Graphics, and others

Evaluate and Purchase Packages on Linux

The four leading software companies other than Microsoft are IBM, SAP, Oracle, and Computer Associates, and they are all enthusiastic supporters of Linux and other open source products:

IBM: most core products run on Linux

SAP: core products, including the my SAP Business Suite, support SuSE and Red Hat Linux

Oracle: core products run on Linux

Computer Associates: offers a full range of enterprise management, data management, change management portal, security, and storage solutions for Linux, including Unicenter and the Ingres database

Cadence: electronic design systems run on Linux, including clusters

PeopleSoft: enterprise business logic and background servers run on Linux

BMC: core products, including MAINVIEW, Patrol, and Deployment Manager, support Linux

SAS: products run on Linux, among many platforms

Verisign: products fully support Linux

Enterprise Application Software

From 1970 onward, there were substantial sales of enterprise packaged software, originally on mainframes.

In the 1990s, there was a surge of growth in sales of a new generation of packaged software.

This was built using client/ server technologies and focused around suites of software for enterprise requirements planning (ERP) and customer relationship management (CRM).

QUERIES

DAY

7

PREVIOUS TOPICS :

Bring New Desktop Systems to the
Underserved

Migrate Applications and Databases to Open
Source

Today's Topics to Discuss:

Five More Open Source Opportunities:

- ❖ Introduction
- ❖ Directory Services
- ❖ Email
- ❖ Groupware and Collaboration

Five More Open Source Opportunities

2.6 Introduction

The opportunities discussed in the previous chapter were:

- Create an open source lab and use it to evaluate and evangelize open source systems.
- Migrate file, print, and network infrastructure to open source systems.
- Build one or more dynamic Web applications using Apache and open source languages.
- Bring new open source desktop and Web systems to the underserved.
- Migrate applications and databases from other systems to Linux.

Customization and Integration

Customization, by **definition**, is the process of making changes to the core of the software system to align with the needs of a business process.

the action or process of successfully joining or mixing with a different group of people

If we are buying a product or group of products as the basis of an architecture that we will extend and integrate, then we are actually buying two things: the products and architecture for development.

Portals, application servers, and integration engines are clear examples of this; any large piece of application software, such as ERP or CRM, is also in practice.

It is becoming increasingly clear that SAP and Oracle, for example, are as much development environments as sets of application products.

Open source software is very suitable to customization.

First, since we have the source code, we can change it or extend it with no questions asked.

Second, there is an opportunity for us to work with the developers to request improvements from them, including sponsoring change

Organization Size

Organizational size is **defined** as the number of employees at any given geographical location. This would include the total corporate **organization** if it is in one geographical location or a division of a decentralized corporation.

Companies vary tremendously in size, and their integration needs and capacity for custom development vary accordingly.

The numbers are, of course, estimates. In particular, businesses such as dotcoms, or financial or technology companies, spend higher percentages on IT.

Larger companies are used to managing multiple environments, often with mainframes, and performing custom development and integration.

Smaller organizations often do not have the staff and budget to do that effectively except in critical parts of the business, and they will emphasize more package acquisition and outsourcing.

Directory Services

A directory service enables directory data to be distributed across multiple servers within a network. While databases are defined in terms of APIs, directories are defined in terms of protocols.

The biggest cost benefit from infrastructure migration often comes from eliminating the directory element from Windows, since the Client Access License (CAL) for directory access is the most expensive software element of file storage.

This also has a benefit in reduction of lock-in, as an open directory will support many different products from many vendors.

The best argument for OpenLDAP is that the directory, touching so many systems and containing your proprietary information, is a very logical place to maintain open standards.

The choices for directory services are the open source OpenLDAP and closed code products, including:

- Novell eDirectory
- Sun ONE Directory Server
- IBM Directory Server
- Microsoft Active Directory

Migration and Interoperability

A **definition** of **interoperability** is "the ability to share information and services".

Human migration involves the movement of people from one place to another

Unfortunately, for many organizations interoperability will be the hardest part of the problem, as usual.

There are tools to migrate directory content on a one-shot basis, and it is not especially difficult to get or build a tool to replicate directory information from a master directory to a copy, as you may do in some cases when introducing Samba to a Windows environment.

Email

Email is the third most common use of Linux servers, after security (firewall and intrusion detection) and Web serving.

A new Linux installation with email is easy to set up. This is not surprising, since email has been a part of UNIX since the earliest distributions.

UNIX Mail Systems

The **Mail command** in **unix** or **linux** system is used to send emails to the users, to read the received emails, to delete the emails etc. **Mail command** will come in handy especially when writing automated scripts.

Migration

Email can have all of the migration problems of a desktop application, including end-user reliance and familiarity, with the migration problems of a complex data center application that has integration and third-party tools issues.

It is high risk, because in modern organizations email is a mission critical application.

The most common email servers on the Internet, according to an analysis by the email vendor Qmail, are:

- Sendmail, 42%
- Other UNIX mail servers, 22%
- Microsoft Exchange, 18%
- Other servers, 18%

PC-Based Mail Systems

The tradition of mail in PC networks is different from UNIX. In the 1980s, when local area networks (LANs) were being installed, mail servers were supplied by the LAN suppliers Microsoft, Novell, and IBM.

The products were originally built around LANs with weaker standards and Internet support and less scalability than UNIX systems, but they were generally easier to administer and use.

Replacing Exchange

It is possible to replace Exchange with all open source products. Exchange itself is an integration of several different functions, and to replace it fully it is necessary to do similar integration. These functions are:

- Message send and receive
- Message store and index
- Global directory (Active Directory) integration for authentication and employee lookups.
- Centralized store for personal information (contacts, calendar, tasks)
- Public folders (simple document management, discussions)

Integrated Exchange Replacements

integration by substitution, also known as u-**substitution** or change of variables, is a method for evaluating integrals.

If you are on Exchange now and want to find a comprehensive product to move to, as opposed to the custom integration solution, there are at least two good choices.

There is an open source product, Exchange4Linux, which is probably a replacement for Exchange in smaller organizations with a few hundred email users.

Groupware and Collaboration

Collaborative software or **groupware** is application software designed to help people working on a common task to attain their goals.

The open source community has been effective at building tools that support the community.

A promising possibility is to look at these community tools to perform a currently unserved function in your organization.

Wiki

The wiki (see <http://wiki.org/wiki.cgi?WelcomeVisitors>), first created in 1995 by Ward Cunningham, allows users to read, create, and edit Web pages on a site using any browser.

This includes not only content, but also organization such as new pages and links.

This is one of the simplest forms of content publishing. Any user can usually participate.

There are now thousands of wikis, built using dozens of different software packages (wiki engines).

Other Community Software

The bulletin board package php BB is a discussion forum, which has been around for several years and is very popular. It is written in PHP and uses MySQL, PostgreSQL, or SQL Serve

Weblogs

Weblogs (“blogs”) are a common method for building community; they went increasingly mainstream this year as the political community adopted them.

There are several choices of software for creating Weblogs. Movable Type, probably the leader, is dual licensed, written in Perl with BDB or MySQL.

Movable Type offers a Web service version, TypePad. Another Weblog product is b2, which is open source software written in PHP with MySQL.

Instant Messaging

Instant messaging (IM) technology is a type of online chat that offers real-time text transmission over the Internet.

A LAN messenger operates in a similar way over a local area network. Short **messages** are typically transmitted between two parties, when each user chooses to complete a thought and select "send".

WhatsApp. WhatsApp is a well-known **instant messaging** app that many users use to chat with friends and colleagues.



QUERIES

?

DAY 7

PREVIOUS TOPICS :

Five More Open Source Opportunities:

- ❖ Introduction
- ❖ Directory Services
- ❖ Email
- ❖ Groupware and Collaboration

Today's Topics to Discuss:

Complex Web Publishing

Manage User Desktops

Other Possibilities

Complex Web Publishing

Complex Web publishing includes content management and portal servers.

Website publishing is the process of uploading content on the internet. It includes: uploading files. updating **web** pages. posting blogs.

Content management (CM) is a set of processes and technologies that supports the collection, **managing**, and publishing of information in any form or medium. When stored and accessed via computers, this information may be more specifically referred to as digital **content**, or simply as **content**.

A web portal is a specially designed website that brings information from diverse sources, like emails, online forums and search engines, together in a uniform way.

The following are examples of portal offerings:

- Plumtree Studio Serve
- IBM WebSphere Portal Server
- Sun iPlanet Portal Server
- Red Hat Portal Server
- Microsoft Sharepoint Portal Server
- PHP-Nuke
- eGroupWare

Portal Components

Any content management portal is going to need all or most of these technologies, either included or integrated: a directory, Web server, application development tools for creating additional custom components, indexing and search, and mail and calendaring.

Usually a product will use the vendor (or its partners) strategic technology for most of these. Microsoft and IBM products use their own directory, database, Web server, development tools, and mail. The Plumtree portal uses the BE WebLogic application server.

Authentication and Personalization

Authentication is the process of verifying the identity of a person or device.

A common example is entering a username and password when you log in to a website.

Entering the correct login information lets the website know 1) who you are and 2) that it is actually you accessing the website.

Personalization is a means of meeting the customer's needs more effectively and efficiently, making interactions faster and easier and, consequently, increasing customer satisfaction and the likelihood of repeat visits.

Content Management

A process must be defined for each type of content to ensure that content can be captured and entered correctly.

This includes information delivered in consistent user-defined hierarchies and other easy-to-navigate structures appropriate to the business. The system specifies how new content gets in.

Indexing and Search

Search can be free-text or structured (based on a classification schema), but it should probably be both.

For most organizations, a specific search should be offered for internal people and locations, and often for products and services.

A **search index** is a body of structured data that a **search** engine refers to when looking for results that are relevant to a specific query.

Indexes are a critical piece of any **search** system, since they must be tailored to the specific information retrieval method of the **search** engine's algorithm.

Component Integration

It tests the interactions between software **components** and is done after **component** testing.

The software **components** themselves may be specified at different times by different specification groups, yet the **integration** of all of the pieces must work together.

As with a daily newspaper, a portal needs both headlines and a rich set of content.

Internationalization

Internationalization is the process of designing a software application so that it can be adapted to various languages and regions without engineering changes.

Localization is the process of adapting internationalized software for a specific region or language by translating text and adding locale-specific components.

Open Source Content Portals

There are a number of products in this area, depending on definition, and in another year it may have changed.

I will select three open source products, which I know to have many users, significant growth and buzz, and architectures that allow rapid deployment of functional applications but scale to support complex custom development when needed.

These are:

- Plone
- PHP-Nuke
- eGroupWare

Plone is a free and open source content management system built on top of the Zope application server. Plone is positioned as an "Enterprise CMS" and is commonly used for intranets and as part of the web presence of large organizations.

PHP-Nuke is a web-based automated news publishing and content management system based on PHP and MySQL originally written by Francisco Burzi. The system is controlled using a web-based user interface.

EGroupware is free open-source groupware software intended for businesses from small to enterprises. Its primary functions allow users to manage contacts, appointments, projects and to-do lists. EGroupware is based on PHP. The project spreads its software under the terms of GNU General Public License.

Plone is developed in Python, using the Content Management Framework and the Zope engine. Python can be used to extend the application.

PHP-Nuke and eGroupWare are both developed in PHP and can be extended with that language.

Manage User Desktops

Linux desktop deployment to a new group of users without existing systems is as simple as any such activity could be. The problem, as usual, is migration.

The only situation that is considered here is that an organization has Windows desktops.

UNIX migration is mentioned elsewhere. The very small group that has anything else at this stage must have its own special circumstances.

Migration of user desktops is a hard thing to do successfully.

First, it involves touching every system, and possibly completely rebuilding it, and this will incur great cost.

Second, users have some say in configuring their systems and have strong likes and dislikes. There is a high risk of user dissatisfaction with this type of change.

Windows desktops could include any combination of Windows 95, 98, ME, Windows NT, Windows 2000, or Windows XP.

They may have Office, again any of several versions, and a great variety of other software.

Migration choices from Windows include:

- Introduction of some open source programs on Windows
- Total replacement of Windows with Linux and open source
- Use of software to make Windows programs available on Linux
- Any combination of these

Analyze Desktop Use and Licensing

We should review software use on the desktop with eventual migration in mind. This certainly includes Microsoft Office (Word, Excel, and Power-Point).

Use should be measured for each product separately even if Office is licensed as one.

Mail, whether Outlook or another, is also an important function to consider. We can also look at use of Access, Project, of course, third-party applications.

The essential data to collect on product use for instance, for the Office products—varies by organization but includes:

- What is the number of licensed users of the product?
- Which version(s) is in use (95, 97, 2000, 2002/XP, 2003)?
- How many users are complex authors, simple authors, primarily readers, or nonusers?
- Are Office documents circulated as part of essential business flows?
- How have prior migrations of desktop software played out in the organization?

Other Possibilities

We can review complex applications for their fitness for open source software over the long haul.

In many cases, we can plan now for systems that will be installed on open source platforms in one or two years from now.

These could be packages, such as enterprise requirements planning (ERP) or customer requirements management (CRM) system, or a custom developed system.

Open Office is an example of a system that was not a particularly successful business on a global scale until adopted by Sun.

There could be similar opportunities in other commercial businesses that could be revitalized through an open source strategy.

They may be candidates for release to open source. Initiatives such as this could be

- Horizontal (e.g., ERP, CRM)
- Industry (e.g., education, electronics)
- Niche (e.g., telecommunications billing, cruise ship reservations)

Telecommunications operates on an extremely large scale.

The billing, rating, and provisioning systems of telecom companies are the largest systems in operation.

Much of the work occurs in near real time, while others are some of the largest batch and paper printing operations around.

QUERIES?

UNIT – III

Operating Systems: Contents of the Operating System- Linux Distribution Vendors- Enterprise Distribution Vendors- Community-Supported Distribution Vendors- International Alternatives. **Open Source Server Applications:** Infrastructure Services- Web Servers- Database Servers- Mail Servers. **Open Source Desktop Applications:** Introduction- Graphical Desktops- Web Browsers- The Office Suite- Mail and Calendar Clients- Personal Software.

Today's Topics to discuss:

Operating Systems

- ❖ Contents of the Operating System
- ❖ Linux Distribution Vendors
- ❖ Enterprise Distribution Vendors
- ❖ Community-Supported Distribution Vendors
- ❖ International Alternatives.

OPERATING SYSTEMS

3.1 Contents of the Operating System

An operating system (OS) contains a kernel, a base operating system, and a considerable number of other packages, where a package is like a product but not sold separately. Whether these additional packages are “included in the operating system” is a commercial and to some extent a political question.

An **Operating System** (OS) is an interface between a computer user and computer hardware. An **operating system** is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.

Kernel is central component of an operating system that manages operations of computer and hardware.

It basically manages operations of memory and CPU time. It is core component of an operating system.

In closed code systems, the vendor decides what is included in the operating system.

The vendor performs integration testing and provides a means for online update.

Further, the packages in the operating system are distributed and supported by the vendor.

FreeBSD

The FreeBSD operating system, whose history was covered previously, is certainly not in widespread use compared with Linux, but it is a stable and powerful operating system.

Although there are less software products available for FreeBSD than for Linux, the most important server products, including Apache and database servers, are available for it.

So FreeBSD is a good basis for a server, particularly a dedicated one. It is used by Yahoo! And by many organizations that need a platform for custom development or manageable Web or database servers.

The Value of Alternative Operating Systems?

It may not be clear why it is valuable to have other operating systems that are not based on Linux.

Linux is stable and successful, is improving rapidly, and is widely used, and I am not advocating that you even consider the open source alternatives.

There has already been one disaster in the open source community: GNU Hurd.

Hurd was supposed to be the kernel of the GNU operating system. It is about 15 years late and still not ready. Fortunately, the Linux kernel was developed.

GNU **Hurd** is the multiserer microkernel written as part of GNU. It has been under development since 1990 by the GNU Project of the Free Software Foundation

Recent Linux Improvements

If you've used any version of UNIX, Linux will be similar enough to be familiar.

However, if your UNIX experience was a while ago, Linux is probably more up-to-date than you expect.

Compared with Windows, it is functionally similar. Many things are done differently, but you should be able to do anything you could do on Windows.

- ❖ OpenOffice with a PowerPoint file
- ❖ OpenOffice with an OpenOffice document file, which I will convert to a Word .doc before sending to the publisher
- ❖ Evolution with a three-pane view similar to Microsoft Outlook
- ❖ Gnumeric with an Excel file

The speed of development in Linux can be quite surprising. In the last half of 2003 there have been improvements in at least these three important areas:

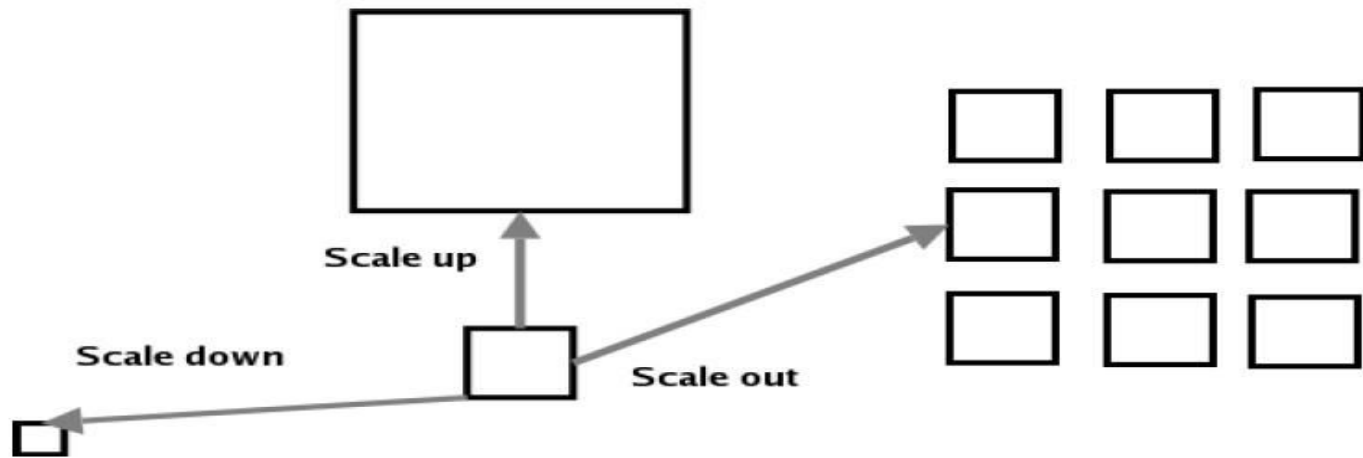
- ❖ Ease of installation
- ❖ Quality of graphical desktop
- ❖ Performance and scalability improvements in the kernel

Scaling Linux up and Down

Scalability is the property of a system to handle a growing amount of work by adding resources ... **Scaling** vertically (**up/down**) **means** adding resources to (or removing resources from) a single node.

Network function virtualization defines these terms differently: **scaling** out/in is the ability to scale by adding/removing resource instances (e.g., virtual machine), whereas **scaling up/down** is the ability to scale by changing allocated resources (e.g., memory/CPU/storage capacity).

Some of the open source systems in place are very large by any measure. There are systems that have very high transaction rates, data volumes, and numbers of users. Some of these systems scale out by using many small systems, in many cases providing reliability through redundancy.



Scaling the operating system

Systems also scale down. Some are consumer appliances, which test extremes at the other end—of cost, small footprint, and time between failures.

There are over a million TiVo systems in consumer use, for example. Other systems run in cellular phones, PDAs, and are embedded in automobiles. Some of these systems can be very inexpensive;

Security

Security means safety, as well as the measures taken to be safe or protected. ... Often this word is used in compounds such as a **security** measure, **security** check or **security** guard.

The **security** department in a business is sometimes just called **security**.

Viruses and worms can be written to run on Linux systems, as any other, and the CERN reporting shows that Linux gets its share of attacks.

Some specific Microsoft decisions for ease of use over security have caused problems, such as running executables from Office and the mail client, but I believe those are mostly fixed now.

Linux Distribution Vendors

- ❖ A distribution vendor does several things, such as:
- ❖ Select the kernel version and apply (and perhaps create) patches.
- ❖ Select a collection of packages with its own versions and patches.
- ❖ Coordinate bug reports and feed them back to package maintainers.
- ❖ Select (and perhaps create) installation and deployment tools.
- ❖ Test and certify the collection and combine onto media for distribution (e.g., physical CDs, ISO CD images, DVD).

The Many Versions of Linux

In adopting Linux, one key decision is the distribution to use. As so often in open source, there are many choices, perhaps more than you would like to deal with.

Several of the distributions offer different editions, such as Professional or Server. Also, of course, there are different versions over time.

In a general sense, the differences are not large, but it can be difficult to make specific statements about exactly how anything works.

For example, take Red Hat. In the last couple of years it has offered Red Hat Linux versions 7.0, 7.1, 7.2, 7.3, 8.0, and 9 plus Red Hat Enterprise Linux 2.1 and 3 and Fedora Core 1 and 2.

Red Hat is undergoing a big one-time transition, and this will resolve to RHEL3 and Fedora.

There are many versions of Windows also. The following versions of Windows were all sold during the last four years, are in widespread use, and are officially supported by Microsoft:

- Windows 98 and 98SE
- Windows ME
- Windows NT4
- Windows CE (several versions, including SmartPhone, Pocket PC)
- Windows 2000 (various editions)
- Windows XP (Professional, Home, Tablet)
- Windows 2003 Server (various editions)

Most-Used Distribution Vendors

At my local users group, Florida Linux Users Exchange (FLUX), a survey is maintained of “favorite” distributions. Among 233 responses were seven distributions, with five over 10 percent, as follows:

❖ Red Hat	28%
❖ Debian	21%
❖ Slackware	15%
❖ SuSE	14%
❖ Gentoo	10%

This group favors the smaller “free” distributions, Debian and Slackware, preferred by many enthusiasts. People may report Slackware or Debian as a “favorite” but use Red Hat when working at their place of employment.

Managing Distributions and Packages

The first choice for an organization to make is whether to purchase an “enterprise distribution” with support or to “roll their own” with a system.

Many organizations may do both, depending on the environment. You can choose among:

- Outsourcing distribution (Red Hat, Ximian Red Carpet)
- Roll your own but use a package distribution mechanism (Debian)
- Outsource your custom package creation (Progeny Platform Services)
- Modifying a distribution to suit, and then distribute it as you wish

Enterprise Distribution Vendors

The two enterprise distributions that are likely to be the choices of larger organizations, particularly those moving to Linux from elsewhere, and particularly in the United States, are Red Hat Enterprise Linux and SuSE Linux (Enterprise and Standard).

Red Hat has been the leading Linux distributor for years, has a high reputation, and has been profitable and growing for several years. SuSE has long been the second major distribution, as well as the leader in some European countries, including Germany.

The name **S.u.S.E.** was an acronym for Software- und System-Entwicklung (Software and Systems Development). The first Linux product sold was an extension of the Linux distribution Slackware, which was delivered on 40 floppy disks.

These two are the likely choice of the organization that is willing to pay for a supported distribution and does not want to build its own.

Community-Supported Distribution Vendors

Debian and Fedora are two major community-supported distribution vendors.

3.4.1 Debian

There are more choices for organizations that want to “roll their own.” The third major distribution is Debian GNU/Linux. It has a very different style: community supported and always free.

Debian is a large distribution that offers a wider set of software choices than any other.

Debian is an operating system and a distribution of Free Software. It is maintained and updated through the work of many users who volunteer their time

Debian, also known as Debian GNU/Linux, is a Linux distribution composed of free and open-source software, developed by the community-supported Debian Project, which was established by Ian Murdock on August 16, 1993.

Debian is a free operating system (OS) for your computer. An operating system is the set of basic programs and utilities that make your computer run.

Debian provides more than a pure OS: it comes with over 59000 packages, precompiled software bundled up in a nice format for easy installation on your machine.

Linux is a type of **Unix**, although purists will tell you that **Linux** is just the "kernel" the lowest-level part. The rest of the OS is made of **Unix** compatible tools from the GNU Software Association and other places. Together, this makes GNU/**Unix**. **Debian** is a general distribution of **Linux**.

Fedora

Fedora creates an innovative, free, and open source platform for hardware, clouds, and containers that enables software developers and community members to build tailored solutions for their users.

Fedora is a Linux distribution developed by the community-supported Fedora Project which is sponsored primarily by Red Hat, a subsidiary of IBM, with additional support from other companies.

Another alternative to the enterprise-branded distributions is the new community- supported version of Red Hat: Fedora Core.

Using Fedora, you will pay essentially nothing to get the distribution but may have to make decisions about support more explicitly than with Red Hat and SuSE.

International Alternatives

Outside the United States there are alternatives in certain areas, notably:

- | | |
|-------------|---------------|
| •Mandrake | Europe |
| •TurboLinux | Asia |
| •Conectiva | Latin America |
| •Red Flag | China |

These have strengths for particular languages and cultures. SuSE, which is based in Germany, is also strong in Europe.

Consumer Linux Choices

For consumers, Linux distributions to consider, in addition to versions of Red Hat and SuSE, include Lindows and Xandros. Xandros, formerly Corel Linux, is based on Debian with the KDE desktop.

Booting from a CD

One more very useful Linux distribution to mention is Knoppix. Knoppix is based on Debian and the KDE desktop with OpenOffice and a powerful set of programs.

KNOPPIX is an operating system based on Debian designed to be run directly from a CD / DVD or a USB flash drive, one of the first of its kind for any operating system.

It comes on a single bootable CD, so you have nothing to install. It recognizes devices, including the network.

Knoppix is great for evaluations or loaner machines, because it just works. It is also a good base from which to fix up Windows or non-booting boxes.

This is very useful for system repair or to provide Linux tools to work with on non-Linux systems.

Knoppix is easy and attractive to use. You can get a good look at a working Linux system and get some work done on it.

Knoppix is at 3.3 and is well tested and popular. Figure 3.3 shows Knoppix in the process of booting from a CD.

It shows the Knoppix screen after booting with the KDE desktop visible, Konqueror open, and Mozilla and OpenOffice available on the menu bar.



QUERIES?

PREVIOUS TOPICS:

Operating Systems

- ❖ Contents of the Operating System
- ❖ Linux Distribution Vendors
- ❖ Enterprise Distribution Vendors
- ❖ Community-Supported Distribution Vendors
- ❖ International Alternatives.

Today's Topics to discuss:

OPEN SOURCE SERVER APPLICATIONS

- Infrastructure services
- Web servers
- Database servers
- Mail servers
- Systems management services

OPEN SOURCE SERVER APPLICATIONS

The important open source server applications, which will be discussed in the following sections include:

- Infrastructure services
- Web servers
- Database servers
- Mail servers
- Systems management services

Infrastructure Services

Infrastructure services consist of basic network services, security services, and file, print, and directory services.

Security services include firewalls, virtual private networking, intrusion detection, antivirus services, authentication, and authorization.

These services are difficult to distinguish at times from basic network services and directory services, which support them, or even mail services, such as the case of antivirus and antispam services.

Infrastructure services are foundational information technologies that are offered as a **service** such that they are supported and managed.

Appliance vendors prefer Linux (or FreeBSD) for two reasons:

- They pay no licensing fees.
- They can tune the system precisely for their needs.

1.File and Print Services

In a mixed environment, we will generally use Samba for file and print services. Linux systems also support file sharing very efficiently and easily using NFS and FTP, and this is a good choice in existing UNIX environments.

NFS, or Network File System, was designed in 1984 by Sun Microsystems. This distributed file system protocol allows a user on a client computer to access files over a network in the same way they would access a local storage file.

Samba allows non-Windows systems to share file and print services with Windows systems. Samba clients function like Windows clients, but for Linux, Mac, or other operating systems, so they see file shares and printers published by Windows or Samba servers.

Samba is an efficient program and scales well.

Companies such as Bank of America and Hewlett-Packard use Samba to support many thousands of clients.

The program, written by Andrew Trumbull while at SGI, is an implementation of the Windows Networking facility called Server Message Block (SMB). The name Samba is a play on SMB.

2. Directory Services

OpenLDAP is based on the original LDAP server, written at the University of Michigan.

It takes a little more work to set up than the commercial alternatives, but it is open source, solid, scalable, and provides authentication that is configurable for many of the services we will want to use:

- Samba file and print sharing
 - Apache Web server
 - Courier and Postfix mail servers
-
- The Mozilla browser and other client programs can read user information from OpenLDAP. In addition, we can program access to OpenLDAP from the command line or from our own applications.

Web Servers

There are really no other general-purpose open source Web servers to consider than Apache.

It has a high share and is the reference standard for a Web server. It is easy to administer and has low overhead, so it works for small sites and systems. The largest Web sites in the world use it.

A web server is server software, or hardware dedicated to running this software, that can satisfy client requests on the World Wide Web. A web server can, in general, contain one or more websites.

1. Apache

Apache is by most measures the most successful single open source software project. It is the most commonly used Web server in the world.

The **Apache HTTP Server** ("http") was launched in 1995 and it has been the most popular **web server** on the Internet since April 1996. It has celebrated its 25th birthday as a project in February 2020. The **Apache HTTP Server** is a project of The **Apache** Software Foundation.

Its job is to establish a connection between a **server** and the browsers of website visitors (Firefox, Google Chrome, Safari, etc.) while delivering files back and forth between them (client-**server** structure). **Apache** is a cross-platform software, therefore it **works** on both Unix and Windows **servers**.

2. Other Web Servers

Alternative general Web servers are the commercial products iPlanet (formerly Netscape) server on various operating systems and, of course, Microsoft IIS, on Windows only.

Oracle **iPlanet Web Server** (OiWS) is a **web server** designed for medium and large business applications. Oracle **iPlanet Web Server** builds on the earlier Sun Java System **Web Server**, Sun ONE **Web Server**, **iPlanet Web Server**, and netscape enterprise **server** products.

- ❖ **Apache Web Server.**

- ❖ **IIS Web Server.**

- ❖ **Nginx Web Server.**

- ❖ **LiteSpeed Web Server.**

- ❖ **Apache Tomcat.**

- ❖ **Node.js.**

- ❖ **Lighttpd.**

Database Server

A **Database Server** is a computer in a LAN that is dedicated to **database** storage and retrieval. The **database server** holds the **Database** Management System (DBMS) and the **databases**. Upon requests from the client machines, it searches the **database** for selected records and passes them back over the network.

Most major databases are available on Linux, and have been for years— Oracle since 1998, for instance. The only major modern database that is not sold to run on Linux is SQL Server.

We can choose to run an open source database, such as MySQL. The open source choice is more likely to deliver significant savings.

Many organizations will be able to use a mixed strategy, combining MySQL and Oracle, for instance, depending on the scale and risk of the application. While open source databases can be used anywhere, they are a particularly good fit where many small databases run, as in a distributed or embedded situation.

1.Classes of Database Servers

We are going to cover all databases here, even desktop ones. We will treat desktop servers as a small class of database server, and some elements of desktop systems as client tools.

So these types of database products need to be looked at separately:

- ❖ Online transaction processing (OLTP) servers
- ❖ Data warehouse servers
- ❖ Embedded databases
- ❖ Client access tools including decision support systems

There are open source choices in all of these areas, but some are stronger than others.

2. Analysis of Database System Sizes

To **estimate** the **size** of a **database**, **estimate** the **size** of each table individually and then add the values obtained. The **size** of a table depends on whether the table has indexes and, if they do, what type of indexes.

The TPC Benchmark

The **TPC Benchmark** (TPC-H) is a decision support **benchmark**. ... This **benchmark** illustrates decision support systems that examine large volumes of data, execute queries with a high degree of complexity, and give answers to critical business questions.

The Transaction Processing Council (TPC) was set up to manage an evolving series of benchmarks starting from TP1 in an independent manner.

The benchmark supports a mix of five transaction types and requires all elements of the database, such as numbers of customers, to scale along with transaction measurements.

Oracle on UNIX supports very large fast databases on several hardware platforms and has many reference cases including the majority of large Internet systems. This specifically includes Linux.

SQL Server on Windows 2000 works with some large reference cases and has the best price/performance by a substantial margin in the size range most customers deploy.

Sybase and DB2 on UNIX work and clearly should work on Linux.

Best choices for very big databases:

Oracle/Linux

SQL/Win2K

DB2/zOS

Best choices for big databases:

Oracle/Linux

SQL/Win2K

MySQL or PostgreSQL

Best choices for medium databases:

- SQL/Win2K
- MySQL or PostgreSQL
- Oracle/Linux

Best choices for small databases:

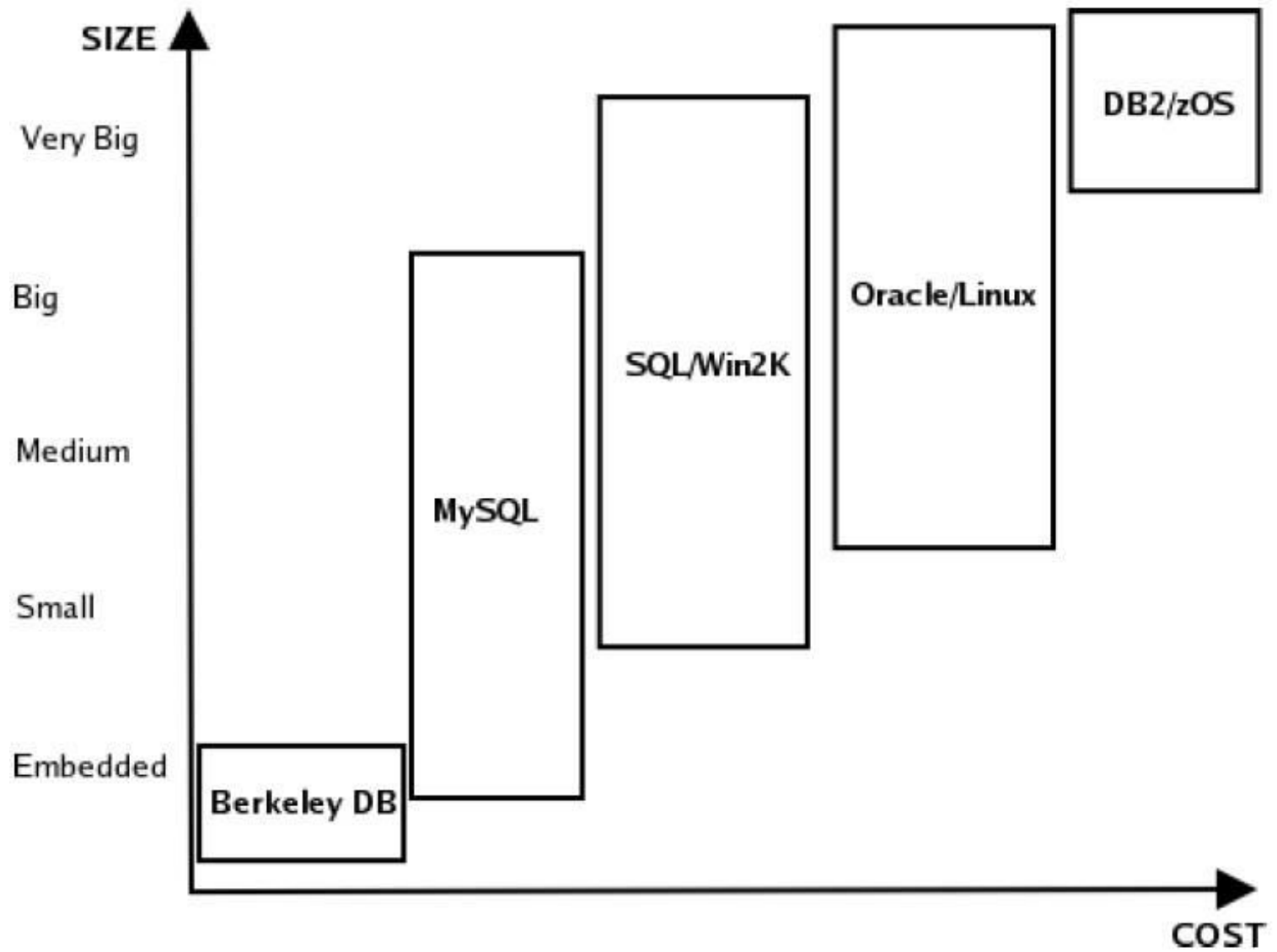
MySQL or PostgreSQL

Embedded Databases

This category includes databases for desktop systems, appliances, and application specific systems that may run on servers or desktops. The performance that matters here is usually fast loading and small memory and disk use rather than transaction rate.

Best choices for embedded databases:

- MySQL
- Berkeley DB



Berkeley DB

Berkeley DB is a core tool under several important open source products, and apparently has 200 million deployments. Berkeley DB (BDB) is a high performance derivative of the old “DBM” databases, which have been part of UNIX and UNIX-like operating systems from the beginning.

PostgreSQL

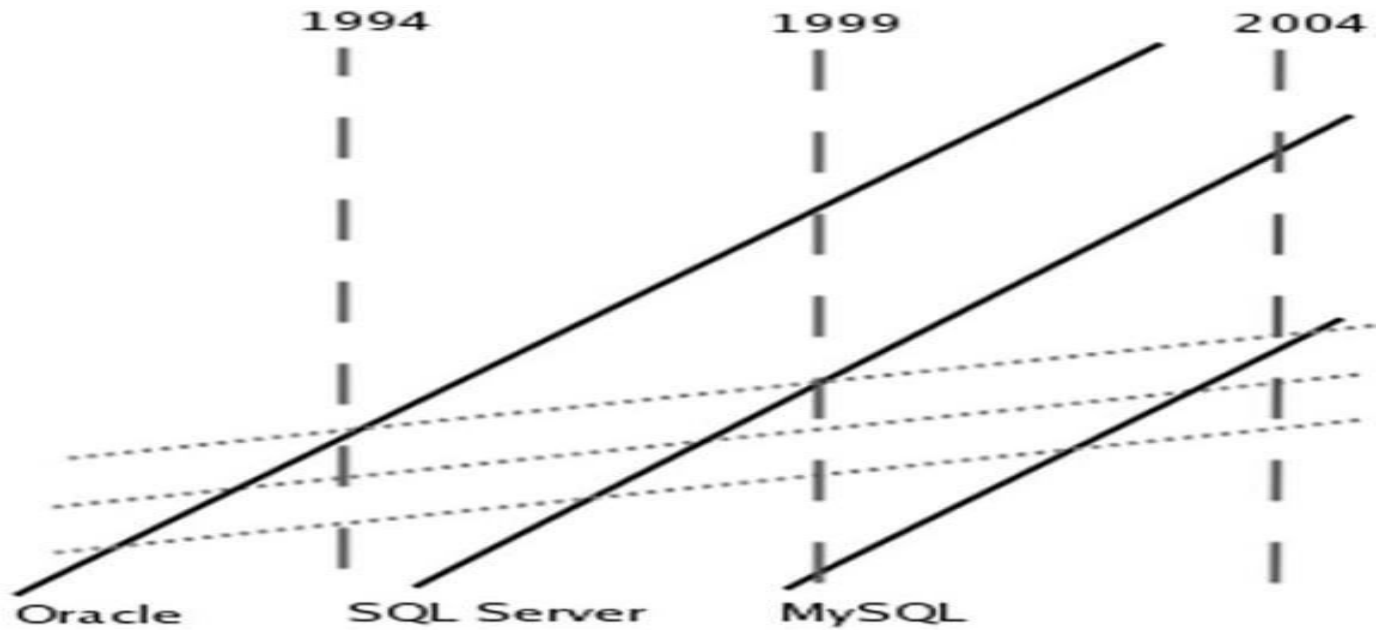
PostgreSQL is an open source database, available under the BSD license / copyright regime. It is based on the Postgres product designed at Berkeley in the 1980s,

MySQL

The MySQL database server is robust, fast, and a very good cross-platform product on clients, including Windows and the Mac and a variety of UNIX servers.

It has a small footprint and good management tools. The product is distributed by the Swedish company, MySQL AB. MySQL is used much more than PostgreSQL; the company estimates about 4M users worldwide.

3. Database Performance Is Good Enough



Mail Servers

A **mail server** (sometimes also referred to an **e-mail server**) is a **server** that handles and delivers **e-mail** over a network, usually over the Internet. A **mail server** can receive e-mails from client computers and deliver them to other **mail servers**. A **mail server** can also deliver e-mails to client computers.

Mail is usually sent with the SMTP protocol and accessed with the POP3 or IMAP protocol. This works about the same with closed code mail servers such as Exchange, but both sending and receiving programs are called Exchange (or Groupwise or Notes).

This is a **list of mail server software**: [mail transfer agents](#), [mail delivery agents](#), and other [computer software](#) which provide [e-mail](#).

System Management Services:

Systems management refers to enterprise-wide administration of distributed **systems** including (and commonly in practice) computer **systems**. **Systems management** is strongly influenced by network **management** initiatives in telecommunications.

The **goal of systems management** is to provide a way for administrators to standarize IT components so that waste and redundancy are made visible and can be eliminated.

QUERIES?

PREVIOUS TOPICS:

OPEN SOURCE SERVER APPLICATIONS

Infrastructure services

Web servers

Database servers

Mail servers

Systems management services

Today's Topics to discuss:

OPEN SOURCE DESKTOP APPLICATIONS

Introduction

Graphical Desktops

Web Browsers

The Office Suite

Mail and Calendar Clients

Personal Software

OPEN SOURCE DESKTOP APPLICATIONS

Open-source software (OSS) is a type of computer **software** in which **source** code is released under a license in which the copyright holder grants users the rights to use, study, change, and distribute the **software** to anyone and for any purpose. ... **Open-source software** is a prominent **example** of **open** collaboration.

Introduction

1. The Open Source Desktop

A complete open source desktop with applications can be easily installed and demonstrated on a typical personal computer using Linux. Most people would agree that such desktops are attractive, powerful, and as easy to learn from scratch as Windows.

Such desktops can be significantly less expensive than closed code systems, since they can save the operating system cost plus the cost of applications such as Microsoft Office.

The important open source desktop applications, which will be discussed in turn, are:

- Graphical desktops
- Web browsers
- Office programs (word processing, spreadsheet, presentation software)
- Professional applications (graphics, database front ends, Web designers)
- Personal applications (media players, games)

2. Linux Desktop Share

Linux has come a long way in power and ease of use, but it is still not widely used on the desktop. Linux has now overtaken since 1994 the Mac in sales to become the operating system on the desktop.

The major computer companies IBM, HP, and Sun all have programs to encourage Linux desktop adoption now, and some major corporate announcements have been made. Linux has exceeded expectations in the past, and may grow on the desktop much faster than currently predicted.

3. Limitations to Desktop Linux Adoption

Whatever Linux growth may be, in the next three or four years we know that there will continue to be an order of magnitude more Windows users than Linux users. This has an effect on the availability of hardware, applications, and support services. Each of these limits the possibilities of Linux desktop deployment significantly.

Hardware

Approximately half of corporate personal computers are now notebooks rather than desktops. It is this group that is least likely to adopt Linux quickly. Setting up Linux on a notebook system is still likely to need some custom work, and there are hardware limitations, including wireless support,

A notebook computer is a [battery](#)- or AC-powered personal computer generally smaller than a briefcase that can easily be transported and conveniently used in temporary spaces such as on airplanes, in libraries, temporary offices, and at meetings. A notebook computer, sometimes called a [laptop computer](#),

Applications

Application software is a program or group of programs designed for end users. Examples of an application include a word processor, a spreadsheet, an accounting application, a web browser, an email client, a media player, a file viewer, simulators, a console game or a photo editor.

Support Services

There are thousands of corporate employees, and many more people in outsourced services, working with users of Windows desktops in support and training roles. Some of these have qualifications such as MCSEs, others do not; but most have a significant investment in the skills needed to support Windows systems and the common applications deployed on them.

Graphical Desktops

It is possible to start and run Linux in a character mode, but this is reserved for installation and debugging situations nowadays. Linux is usually installed to start in a graphical mode, running a windowing system, and a desktop system will usually run a desktop manager with a set of integrated utilities.

In [computing](#), a **desktop environment (DE)** is an [implementation](#) of the [desktop metaphor](#) made of a bundle of programs running on top of a computer operating system that share a common [graphical user interface](#) (GUI), sometimes described as a [graphical shell](#).

The desktop environment was seen mostly on [personal computers](#) until the rise of mobile computing. Desktop GUIs help the user to easily access and edit files,

Essentially all Linux systems use the X Windows system (X11) as the graphical user interface (GUI)—generally XFree86, which is the most used port of X11 on Intel.

X11 was written at MIT in the early 1980s and made available under an open source license similar to Berkeley.

The X Window System is a windowing system for bitmap displays, common on Unix-like operating systems. X provides the basic framework for a GUI environment: drawing and moving windows on the display device and interacting with a mouse and keyboard.

The X11 toolkit is written in the C language and accessed at quite a low level, so it is not easy to program for. “Widget sets” were developed to make this development more consistent and less low level; these evolved into object-oriented APIs, and there are several of these.

So Linux systems have a common graphical code base, but alternative sets of window managers, look and feel, and development toolkits.

For general business systems, we will usually prefer a standard desktop, such as we get with Windows or the Mac, since this can reduce costs of training and support. Standard desktops provide a combination of useful components:

Window manager, which organizes windows, menus, and scroll bars in a coherent manner

Desktop arrangement, or a place to launch programs, organize and find things

File manager, similar to Windows Explorer or Mac Finder, which may be integrated with a Web browser

Useful applications

Games

This is the type of package of components that is offered by Windows or the Macintosh. Of the many desktops available for Linux, two are of major importance. They are the Gnome and KDE projects, which both started around 1997.

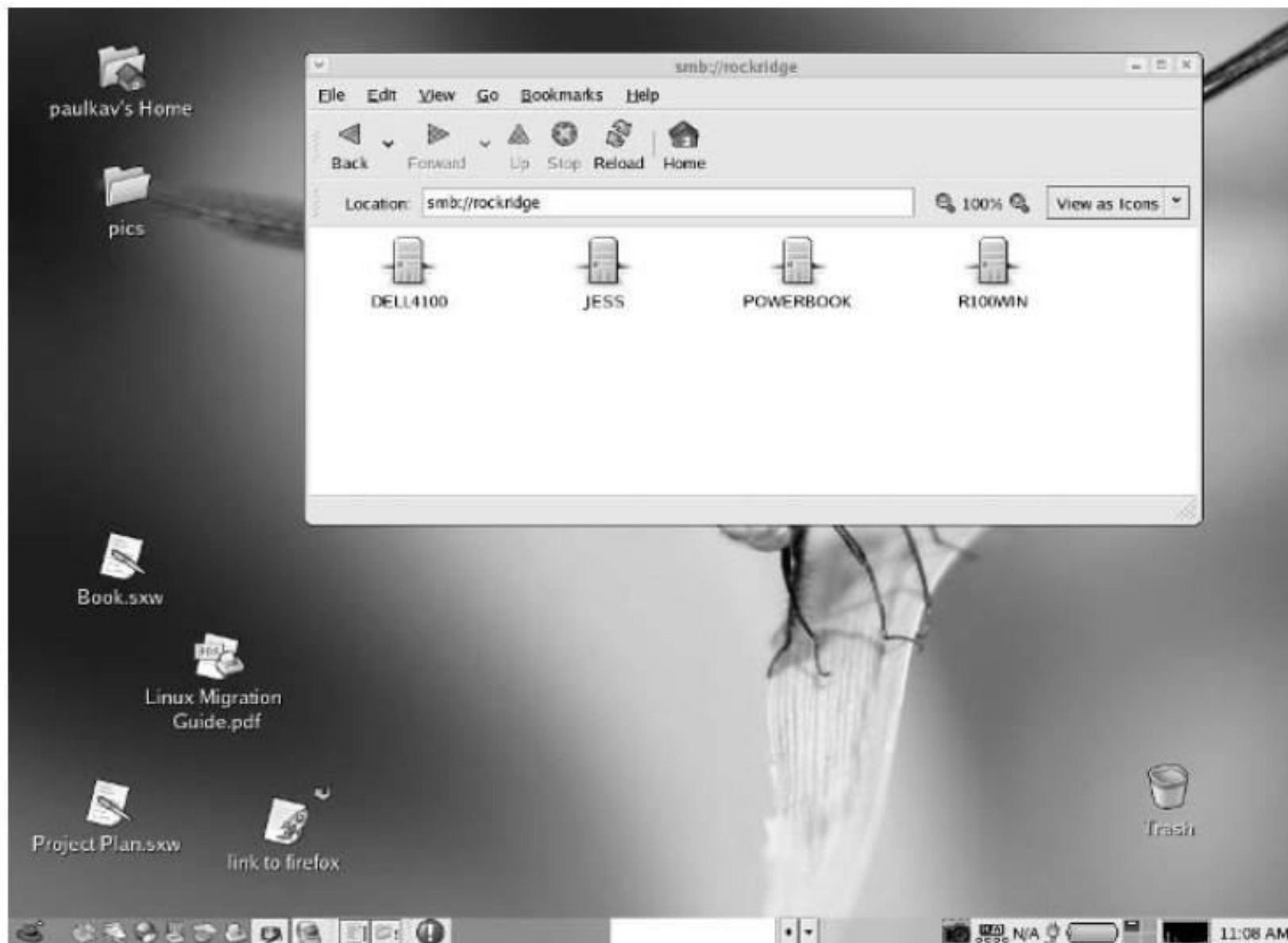
GNOME is a free and open-source desktop environment for Unix-like operating systems. GNOME was originally an acronym for GNU Network Object Model Environment, but the acronym was dropped because it no longer reflected the vision of the GNOME project.

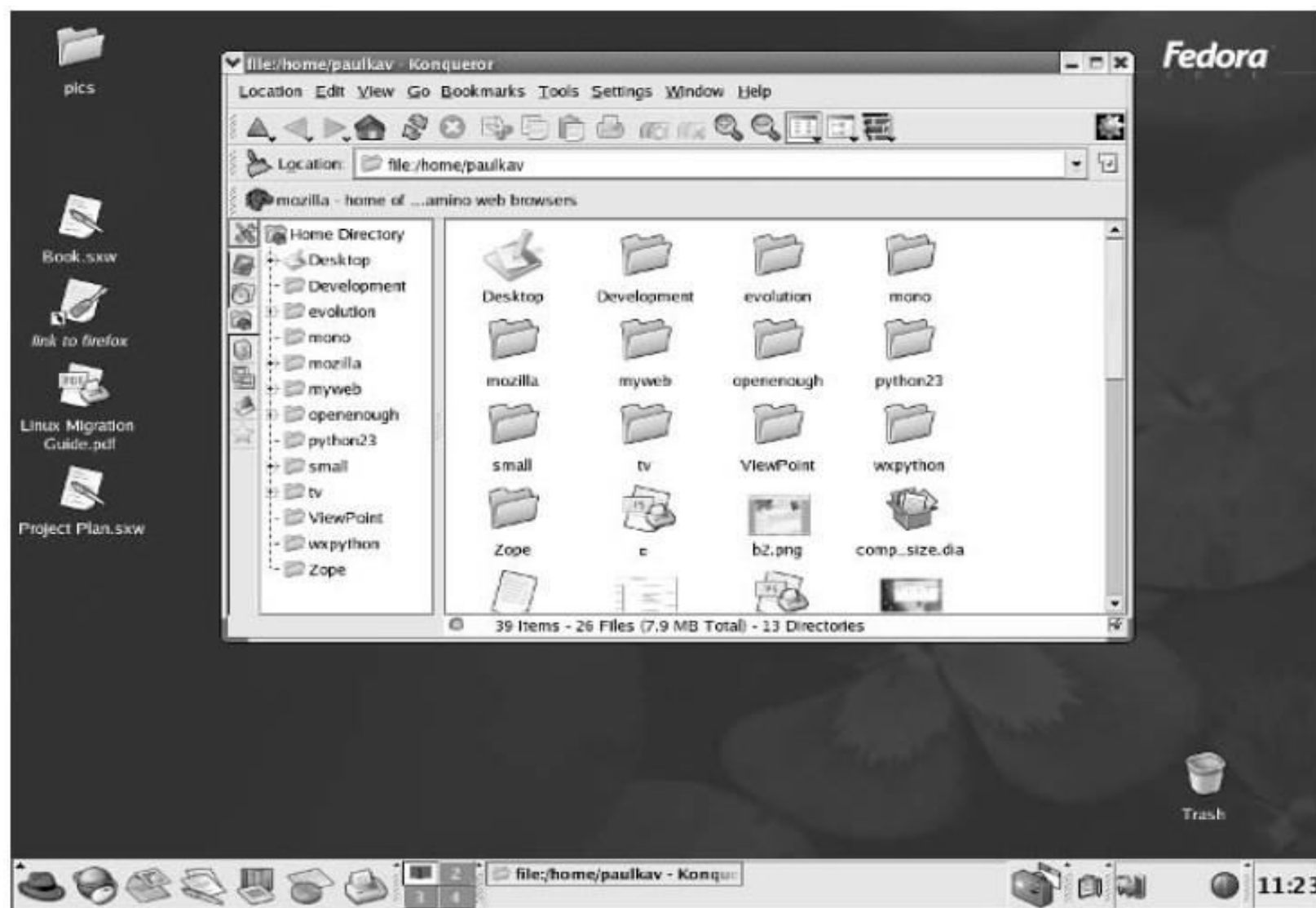
KDE is an international free software community that develops free and open-source software. As a central development hub, it provides tools and resources that allow collaborative work on this kind of software.

- The Gnome desktop shows:
- A graphic saved as background
- The Home folder at the top left
- Trash at the bottom right
- Other folders and links on the desktop
- The file manager looking at some network file shares

The Gnome menu at the bottom, including the following from left to right:

- ❖The main menu (Red Hat) button
- ❖Application launchers for several applications
- ❖A screenshot utility
- ❖Wireless and battery information
- ❖The virtual desktop manager
- ❖System manager utilities
- ❖Clock





The KDE desktop is similar, but has a different graphic background and is running the Konqueror file manager, not Nautilus. The menu bar is similar. It happens to be configured a little larger and with some items placed differently.

Web Browsers

When developing applications, we do not usually want to require a particular Web browser and operating system. In many cases, we cannot know which browser an application user will be using.

In principle, we know that if too many people write to a specific browser that could destroy one of the best features of the Web, its independence from particular hardware and software platforms. Generally, the best plan is to write for several browsers.

Browser	Platform	Status	Comment
Mozilla, Navigator	Windows, Linux, Mac	Open source	Navigator is based on Mozilla 1.6; Mozilla Firefox is new code, much smaller.
Internet Explorer	Windows	Included with OS	On older Macs but now discontinued.
Konqueror	Linux	Open source	Uses Qt libraries, works well with KDE.
Safari	Mac	Included with OS	Based on the Konqueror rendering engine (Qt runs native on the Mac).
Opera	Windows, Linux, Mac	Free with advertising, or fee	Small, fast, but not open source.

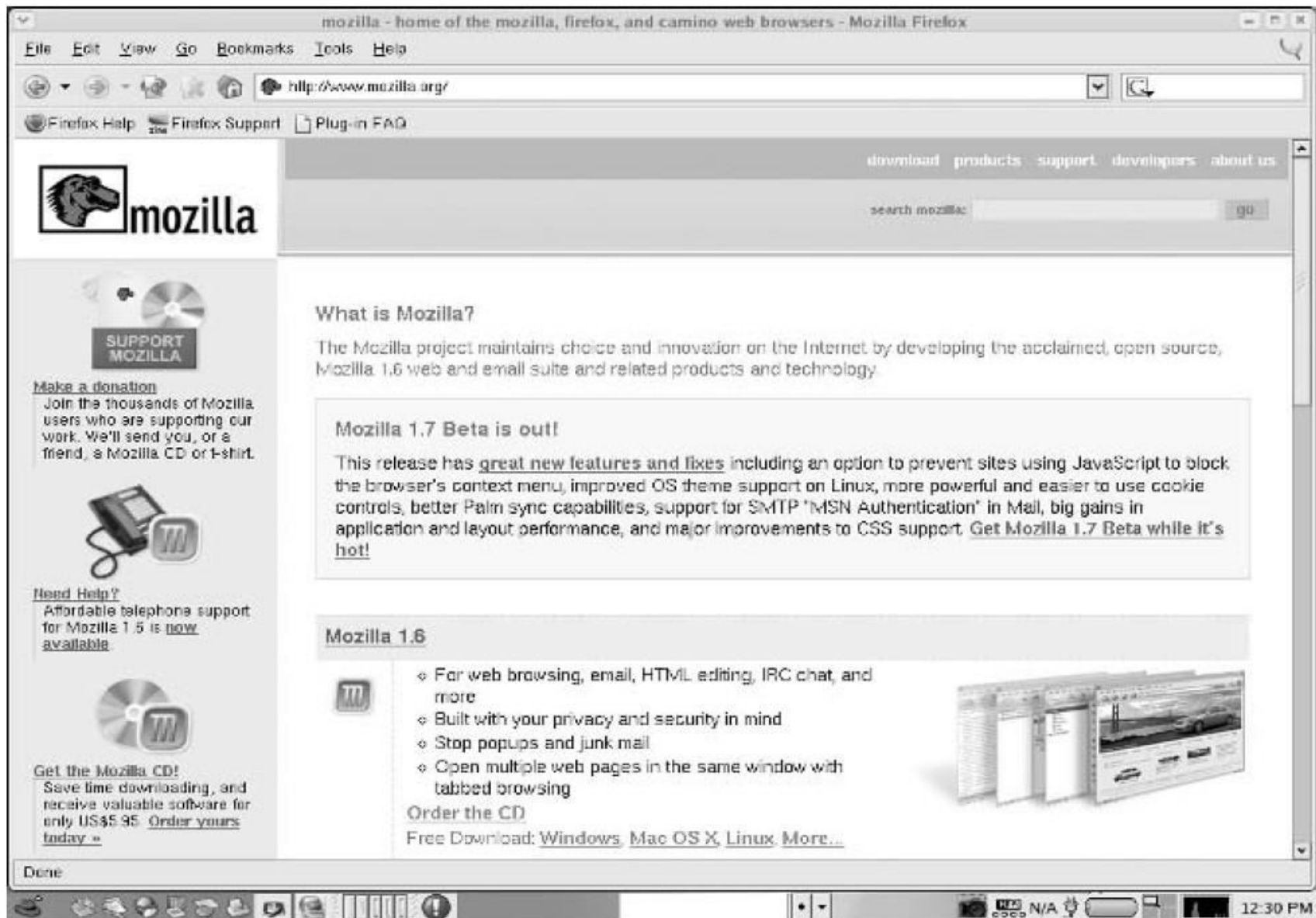
1. Deploying Browsers

Although we may want to support several browsers when developing, when deploying desktops .

Most Windows shops use Internet Explorer (IE) for obvious reasons: It is good enough and is already installed. The limitations of IE, such as its lack of control over pop-ups, can be addressed with third-party add-ins or managed from the firewall.

Organizations that would like a single browser across multiple platforms can select the open source Mozilla, either Firefox or the older integrated versions, Netscape or Opera.

Once a browser is installed, the common plug-ins needs to be installed. The four most commonly used are Adobe Reader, the Real Audio Media Player, Macromedia Flash, and the Java 2 run time.



2. The Office Suite

In considering the office suite, we will consider the word processing, spreadsheet, and presentation programs, although the open source suites, such as Microsoft, include other programs, such as drawing and image management. In terms of Microsoft Office, then, we are looking at replacements for Word, Excel, and PowerPoint.

There are several alternative open source office suites:

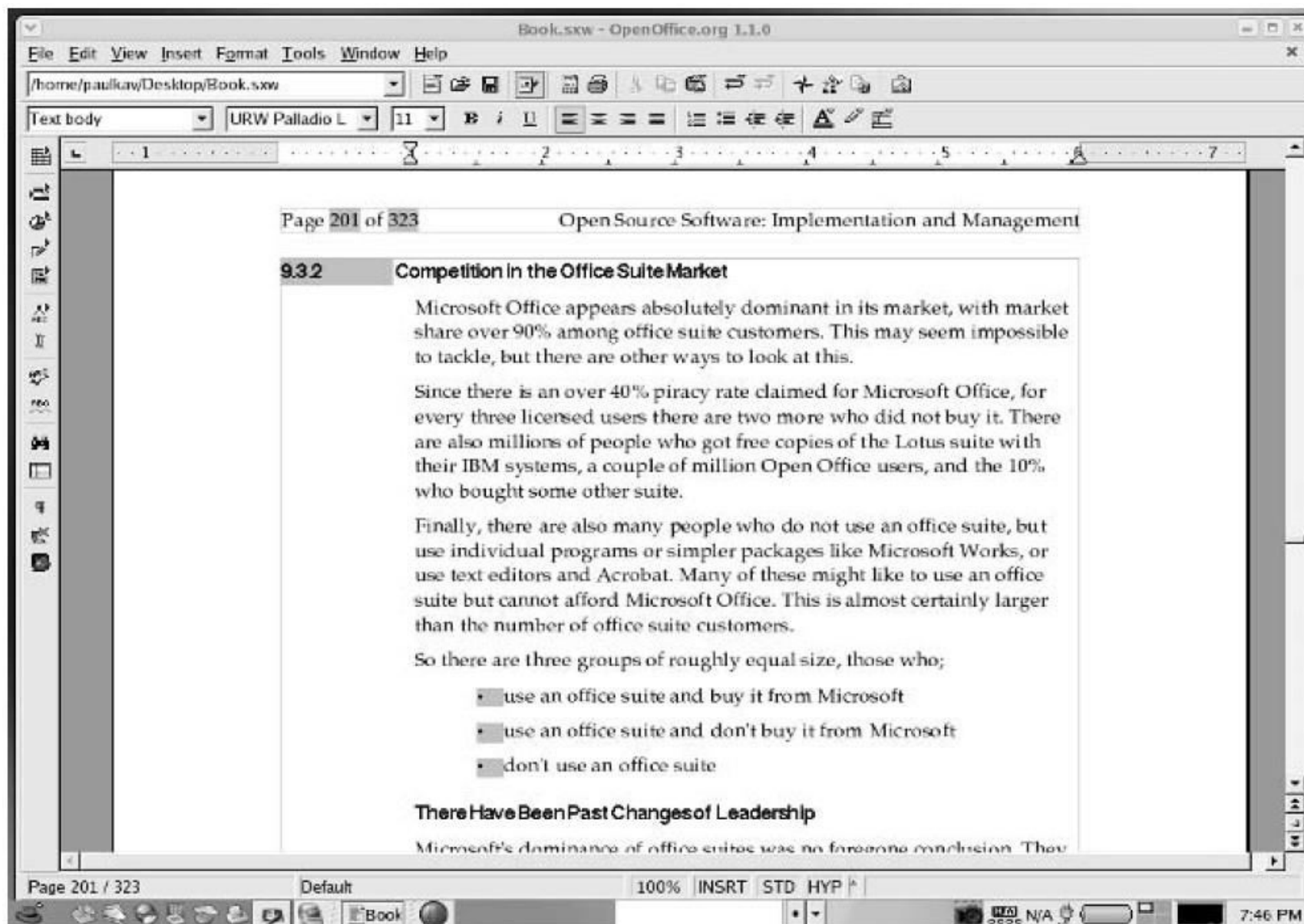
OpenOffice

KOffice

Gnome Office

OpenOffice.org

OpenOffice.org (abbreviated here to OpenOffice) is the leading open source office suite. Sun purchased StarDivision, the German developers of StarOffice, in 1999.



3.Comparison of Microsoft Office to OpenOffice *Bundling*

OpenOffice does not include an email client like Outlook, but most people will use Evolution, which is powerful, similar to Outlook, integrates well with OpenOffice, and is open source. Similarly, OpenOffice does not include a database program, but most people will consider MySQL if they need a SQL database program. MySQL is more powerful and scalable as a database than Access,

1. Integration

2. Formats

Microsoft Office cannot read OpenOffice formats at all, while OpenOffice reads Microsoft formats quite well.

OpenOffice can create PDF files directly, while Microsoft Office has to use a third-party tool.

4. Migration from Microsoft Office to OpenOffice

The Microsoft installed base is Office 97 and Office 2000. There is very little Office XP or Office 2003 yet. This immediately highlights the main problem with Office migration today, which is that nobody wants to do it.

Importing/Exporting between MS Office and OpenOffice

First off, Microsoft Office cannot read OpenOffice files at all. Any OpenOffice files must be converted to Microsoft formats in OpenOffice. OpenOffice-specific features will be lost in these formats.

OpenOffice can read and write Office documents and templates in the 97/2000/XP format.

5 Lock-in and Complexity

Not all organizations will be able to migrate away from Office now. It depends on the way they use it. Users will be slower to change if they are locked in (something store), because they use Microsoft Office features that do not migrate. How seriously your organization is locked in needs to be evaluated for each group of users in the organization. It is a function of:

- Advanced or professional users authoring documents
- Use of technical features (macros, shared components, etc.)

The overall pattern of collaboration over documents

1. ***One-Time Migration*** (moving data from **one** location to another)
2. ***Two-Way Interoperability*** (**Two** or more different systems or parts of systems can exchange and use information readily)

Mail and Calendar Clients

There are several good email open source clients available. This includes browsers that also do mail, such as Mozilla and Opera, and dedicated email clients, such as Eudora.

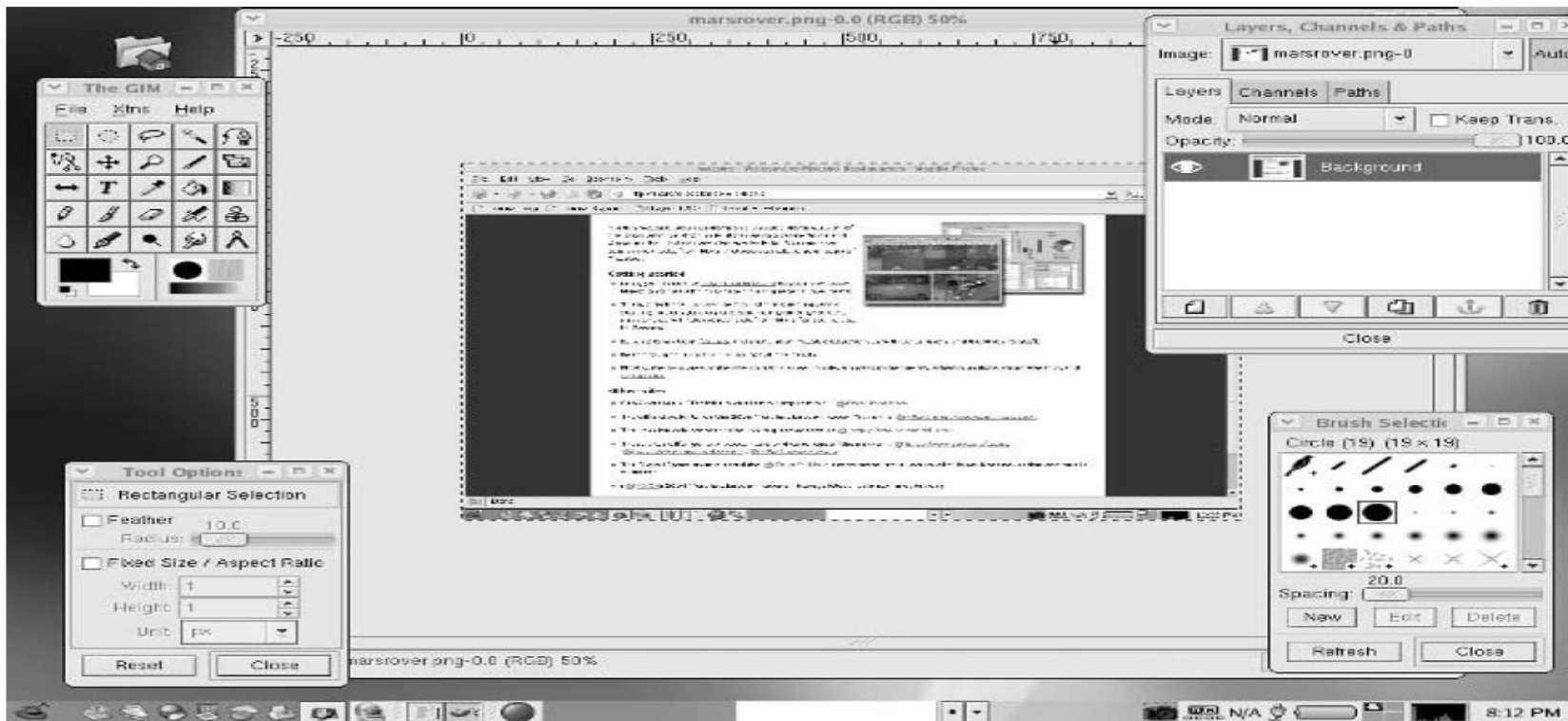
A big question with mail programs is the extent to which you want to replicate Outlook. If you want the Outlook features, including the bundling of calendar, email, and small-scale personal databases, you will probably want to use Evolution, which matches the look of Outlook very well.

1. Professional Applications

This includes applications for project management, drawing and image management, and other professional work. In some ways the situation is similar to that with Office. There are good open source programs available, but they may not match feature for feature, and migration raises problems of data formats and user training.

2. Drawing and Image Management

The open source programs GIMP compare favorably for general users with PhotoShop, Visio, and Illustrator. As with Office, the most demanding professional users will not switch because of their time invested and high-end needs.



Personal Software

This area is one where Linux is catching up quickly. I still think that today if you simply want to choose the best machine for performing multimedia functions or games, no other considerations, you should look at the Macintosh or Windows XP.



1. Running Windows Applications

Sometimes we have to run an application that is not available on Linux. Most needs can be met in a general way, but there are quite often particular programs that are not available. If it is necessary to run a particular program that is not available on Linux, this can be met with a variety of techniques.

QUERIES?

UNIT – IV

How Open Source Software Is Developed: Methodology- Languages Used to Develop Open Source Products- Cross-Platform Code. **Managing System Implementation:** Implementation Roles- Open Source Impact on Team Issues- Implementation Process- Implementation Principles- Key Documents- Migration- Interacting with the Open Source Community.

❖ **Today's Topics to discuss:**

❖ **How Open Source Software Is Developed:**

- ❖ Methodology

- ❖ Languages Used to Develop Open Source Products

- ❖ Cross-Platform Code.

❖ **Managing System Implementation:**

- ❖ Implementation Roles

- ❖ Open Source Impact on Team Issues

- ❖ Implementation Process

HOW OPEN SOURCE SOFTWARE IS DEVELOPED

4.1 Methodology

In Open Source there are several models that are used in use. These include:

- An individual working largely alone
- A conventional collocated product team

1. Open Source Compared with Closed Code

If we compare successful open source development to the successful development of similar closed code software packages—for example, at Microsoft—the similarities appear to be larger than the differences.

Estimates on the number of open source developers vary, but cluster around a million. Most of them work some of the time in closed code development.

Great open source and great open source projects usually have great developers in charge. In closed code software companies, these three methods (giant group, product team, individual) all have their counterparts.

Comparing similar product development efforts such as Linux to Windows NT, MySQL to SQL Server, PHP to ASP, or Python to Delphi/C# we usually see similar project structures, time frames, and leadership.

2. Open Source Compared with Corporate Development

Comparing open source development to corporate development, we do see much bigger variations.

First, both open source and closed code software developers pay a very high price for failure. Corporate developers can fail and live again, and analysis shows that most corporate systems do fail on measures of time, cost, and quality. So we are really measuring successful software products against all corporate practices, mostly not successful.

Second, corporate development is not usually maintained over the same time frames and with the same levels of staffing and consistency as open source.

Third, while the software house toolkit is similar to open source the corporate developer favors proprietary toolkits that have shifted over time, currently usually Java based.

3 .Open Source Development Tools

The open source community is conservative and frugal in tool use. Open source tools include CVS, and other text editors; GNU Make, debugger, and other tools. All these tools have a long tradition and are very effective.

4. Managing People

As with the research scientist, the software engineer in a commercial world often has a culture conflict with the salespeople and others in the corporate end of the business.

Motivation in the software community includes:

Playing with technologies and experiencing the sheer fun of writing code

Getting something done that an individual needs, such as inventing the Perl language, or a peer-to-peer music sharing system

Earning money, or enhancing career and skill development (which is deferred money)

Languages Used to Develop Open Source Products

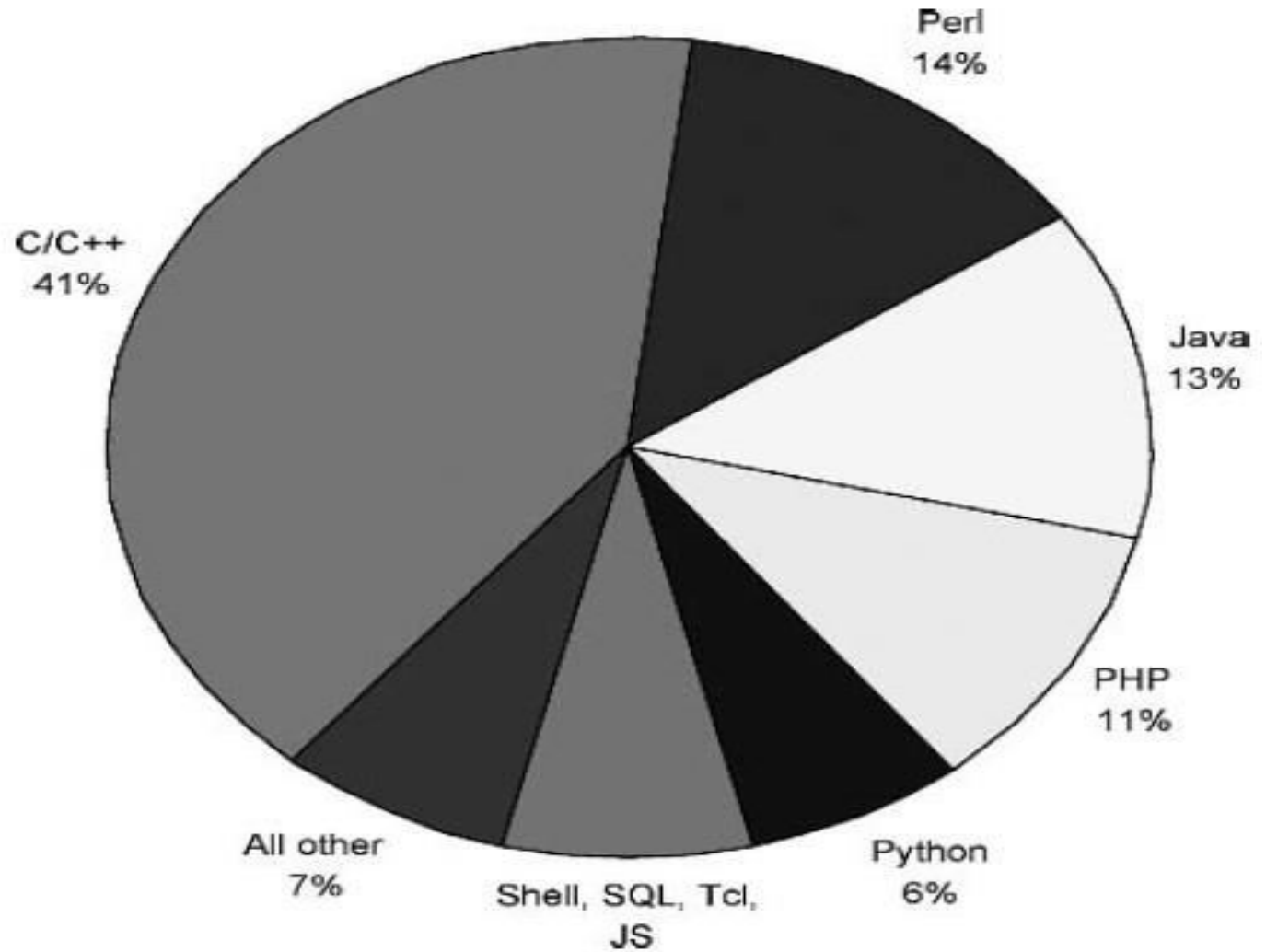
It is interesting to analyze the languages used to create open source software. This in three ways:

Analysis of data on language on SourceForge (<http://sourceforge.net>), which is by far the largest repository for open source projects.

SourceForge is a web-based service that offers software developers a centralized online location to control and manage free and open-source software projects.

Review of the best-known open source programs, such as Apache

Anecdotal review of developers at open source conferences and user groups



Open source language use

Language	Number of Projects
C/C++ (C 5639, C++ 2550)	8,189
Perl	2,806
Java	2,473
PHP	2,124
Python	1,238

1 .C and C++

Half of the projects on Fresh meat are C or C++, with the rest split fairly evenly among the higher-level languages: Perl, Java, PHP, and Python. Most of the successful open source products so far are written in C/C++.

C++ is a general purpose programming language and widely used now a days for competitive programming. It has imperative, object-oriented and generic programming features. C++ runs on lots of platform like Windows, Linux, Unix, Mac etc.

2 .Perl

Perl is a family of two [high-level](#), [general-purpose](#), [interpreted](#), [dynamic programming languages](#). "Perl" refers to Perl 5, but from 2000 to 2019 it also referred to its redesigned "sister language“.

Perl was originally developed by [Larry Wall](#) in 1987 as a general-purpose [Unix scripting language](#).

Perl is a good language for sophisticated scripting. IT is criticized for being hard to understand and is probably a better choice for individuals than teams.

Perl applications include several system administration tools, including Webmin, the very popular SpamAssassin tool, and the Movable Type Weblog tool.

3. PHP

PHP is a Web development language similar to Active Server Pages and Java Server Pages. PHP originally stood for Personal Home Page; it was more like Front Page originally but has evolved into an object-oriented language.

PHP is primarily a Web language, but it can be used for scripting, and there is a version of PHP for GUI development called PHP-GTK.

PHP is powerful, an easy language to learn, and there are several well rated open source applications written in it. PHP appears to be the leading language for developing Web applications

4. Python

Python is an object-oriented language that is easy to learn and use. It has extensive libraries available to allow development of pretty much any kind of application, including games, OpenGL graphics, cross-platform GUI, and network programming. As a general-purpose language Python is powerful, balanced, and well organized.

5. Java

Java is a [class-based, object-oriented programming language](#) that is designed to have as few implementation [dependencies](#) as possible. It is a [general-purpose](#) programming language intended to let [application developers](#) *write once, run anywhere* (WORA), meaning that [compiled](#) Java code can run on all platforms that support Java without the need for recompilation

Java is less of a general-purpose scripting language with built-in functions for common tasks like Perl or Python, but it does have a strong available set of classes

Cross-Platform Code

In computing, **cross-platform** software (also **multi-platform** software or **platform-independent** software) is computer software that is implemented on multiple computing **platforms**.

Most servers today run Windows, Linux, or another form of UNIX, and larger organizations have a mix of those. Those servers today are split fairly evenly between the UNIX-like systems (including Linux) and Windows.

Since there will be a long period where most organizations will have Windows and Linux servers, it is not necessary for server software to run on both systems as long as it can interoperate.

There is an opportunity cost to making applications run on multiple platforms..

Product	Function	Competes with	Runs on
Apache	Web server	IIS	Windows, Linux, UNIX, others
Oracle	Database server (client tools)	SQL Server	Windows, Linux, UNIX, others
Notes	Calendar & groupware	Exchange	Windows, Linux, UNIX, Mac, others
Opera or Mozilla, Netscape	Browser (client)	Internet Explorer	Windows, Linux, Mac, others

Why Open Source Is Cross-Platform

Open code developers are able to develop cross-platform. The languages that are used for open source (C++, Perl, Java, PHP, Python) are all available on Linux, Windows, and other UNIX platforms, including the Mac. So are key elements of the development platform: shell, database, Web server, support tools.

Open Source on Cross-Platform in Practice

Looking at open source products in actual practice shows that most of the more popular server products are available on Linux (all distributions), Windows, and some forms of UNIX, with a few on Linux/UNIX only—for example, Apache, MySQL, PHP, Perl.

Popular client products (media players, instant messaging, office, mail) will mostly be Linux/Windows/Mac but may be Linux/Mac or Linux only.

MANAGING SYSTEM IMPLEMENTATION

4.4 Implementation Roles

A software development or installation team is generally a very mixed team, regarding skill types and levels and often other factors, but it runs as a team of peers.

There may be varying talent and experience, but good teams tend to be highly demanding of talent and work effort.

Teams must also work well with other groups in the company, which may have different skills and motivations, and must interact externally with customers, suppliers (such as software component and tool vendors), competitors (to understand the market situation), and other external factors such as regulation and funding availability.

Teams vary in size, but some components are always present. One way to approach this is to define a set of roles. We can define six internal roles as follows:

Customer management

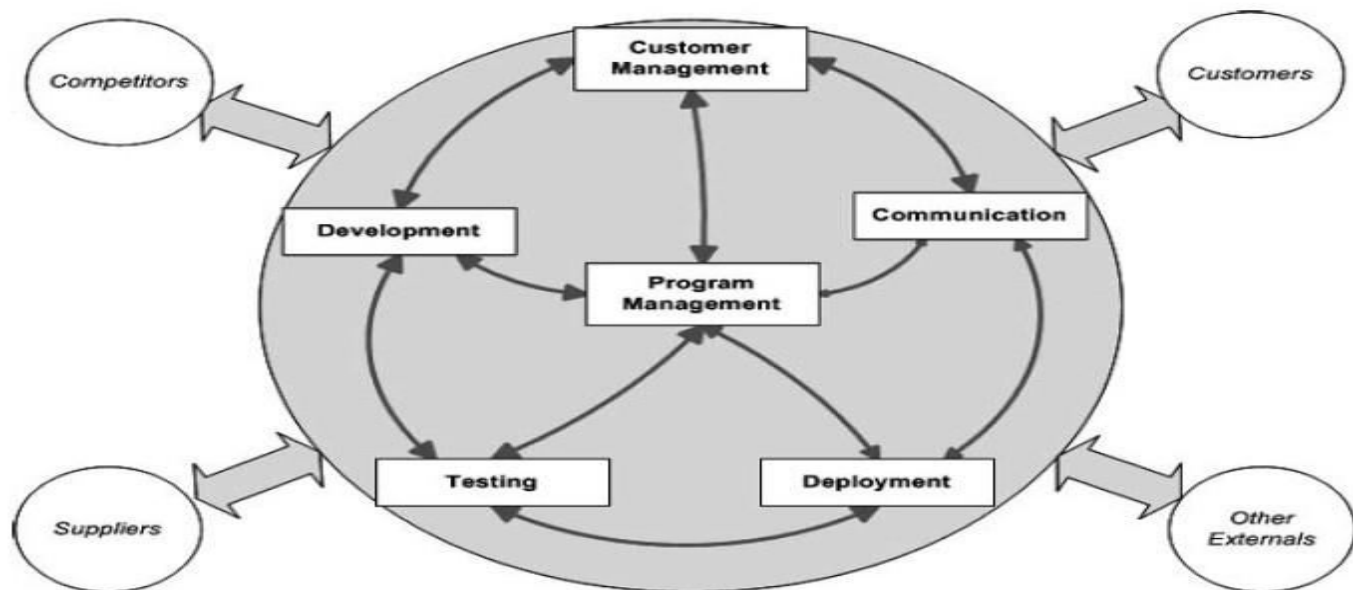
Program management

Development

Testing

Communication

Deployment



1.Program Management

The program management role manages and maintains the project plan and schedule. It is responsible for managing activities in line with cost and personnel plans. It maintains and updates the risk estimates.

2. Development

This role designs and builds the code and tests it at the unit level. The development role selects the tools used in development and the components incorporated into the product. The success of this role is measured by the delivery of working product and the rate of addressing reported issues.

3. Testing

This role involves translating the specification into functional test plans, selecting and creating automated tests, reporting and tracking problems, managing the testing community to ensure test coverage, and managing the product toward shipping.

4. Communication

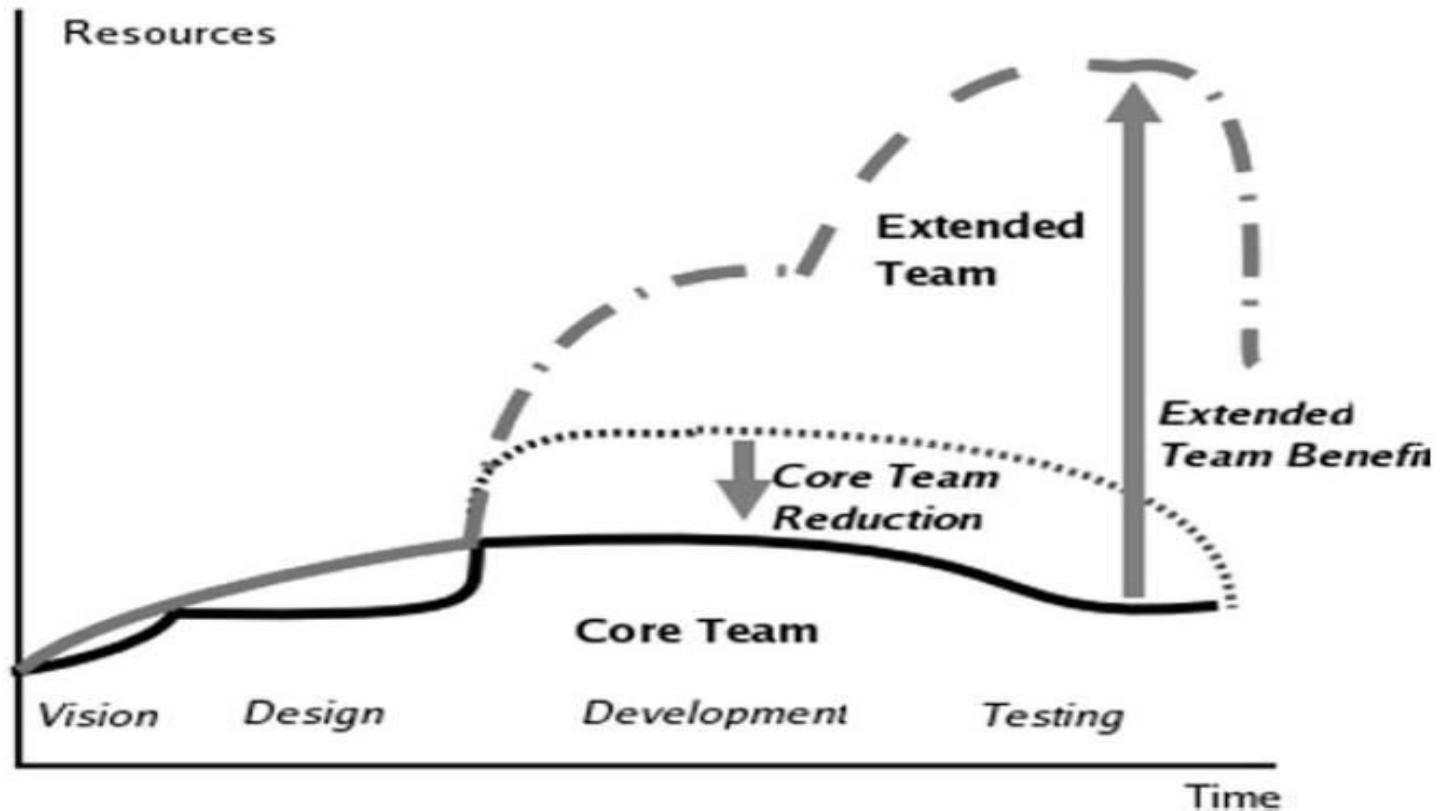
This role manages communication with the user, including interface design, documentation, a help system, and training materials. Documents may be paper or interactive; these days they are often composed of HTML and may include multimedia elements. The success of this role is measured by user uptake, satisfaction, and level of complaints.

5. Deployment

This role manages system setup and delivery. This may involve researching particular problems with remote sites and special client situations. The success of this role is measured by issues at deployment and after. This is a key channel of communication with customers.

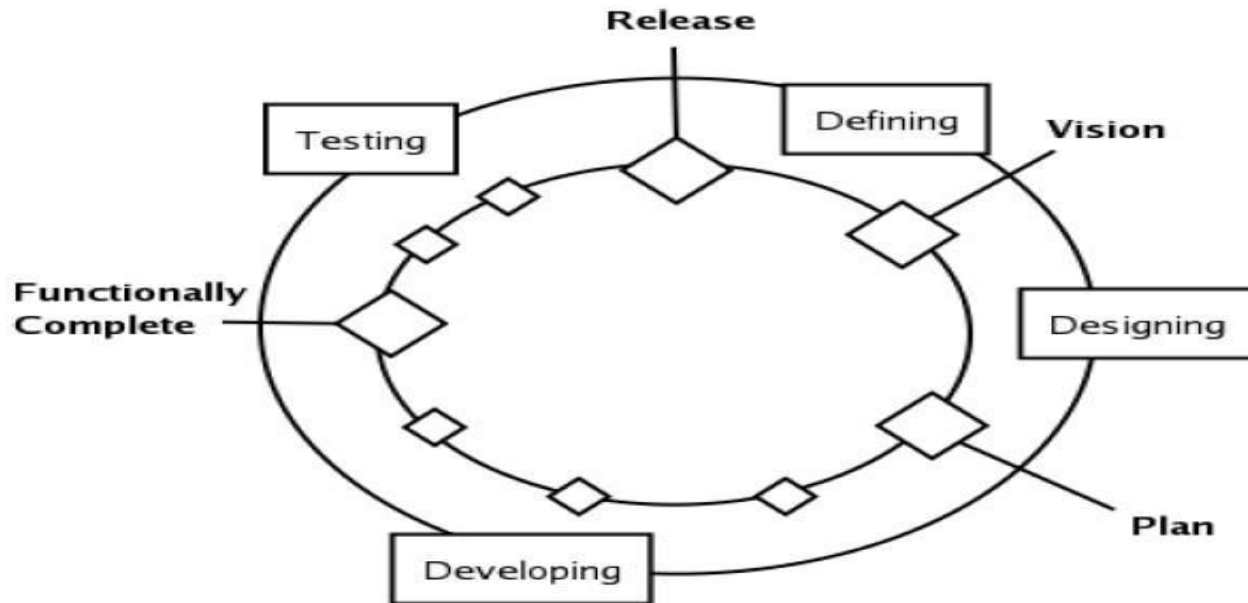
Open Source Impact on Team Issues

Development moves in increasingly rapid cycles, and this is continuing. This makes incremental development and use of rapid languages increasingly necessary. There is a trend to more types of outsourcing, so that teams are more virtual and probably have little in common except the specific project.



Implementation Process

Each release contains four phases: defining, designing, developing, and testing. Table 4.6 lists the phase definitions.



Development process model.

Defining	<p>The team defines goals and determines scope.</p> <p>Interim milestones are team formation and drafts of the project definition components.</p> <p>End milestone is delivery of the project definition, complete and reviewed.</p>
Designing	<p>The team drafts a specification and plans a schedule.</p> <p>Interim milestones are design components, reduction of high risks, and more complete costs and schedule.</p> <p>End milestone is delivery of the functional and technical specs and updated project definition, complete and reviewed.</p>
Developing	<p>Interim milestones are partial releases and components.</p> <p>End milestone is a functionally complete system with managed source code and executables, user support and testing support materials, and updated functional and technical specs and project definition.</p>
Testing	<p>Interim milestones are typically preliminary releases. This is driven by problem reporting, until the number of acceptable severity bugs allows release.</p> <p>End milestone is release.</p>

1. Releases

There can be some overlap between phases, but generally this does not include coding, which needs special arrangements. Most commonly, the vision and plan for the next release can take place during testing of the current release.

Releases iterate at two levels. In addition to external releases, there will often be releases at an internal level.

2. Team Roles during the Process

During the phases, team roles vary to some extent. Table 4.7 shows the focus of different roles during different phases.

Role	Defining	Designing	Developing	Testing
Customer management	Owns process; vision document; concept	Communication plan; managing customer expectations	Manage customer expectations; communicate with customer	Communication; test site/result coordination; next release planning
Program management	Design goals; project risks and costs	Owns process; functional spec; project plan	Track project; communicate with team; plan testing	Project and bug tracking
Development	Prototypes; feasibility; Approaches	Development plan; technical spec	Owns process; build and test	Bug resolution; optimization
Testing	Acceptance criteria; testing strategies	Evaluate design for testability; detailed test plan	Develop test cases and scripts; test components; begin testing releases	Owns process; bug finding and reporting
Communication	User performance and communication needs	Plan for user performance and education	Develop and test	Training; bug fixing
Deployment	Major deployment and support considerations	Evaluate design for deployment; deployment plan	Operational documentation; Plan staging; support internal deployment	Test deployment; deployment planning; operations/ support training

QUERIES?

PREVIOUS TOPICS:

❖ How Open Source Software Is Developed:

- ❖ Methodology

- ❖ Languages Used to Develop Open Source Products

- ❖ Cross-Platform Code.

❖ Managing System Implementation:

- ❖ Implementation Roles

- ❖ Open Source Impact on Team Issues

- ❖ Implementation Process

Today's Topics to discuss....

- ❖ Implementation Principles**
- ❖ Key Documents**
- ❖ Migration**
- ❖ Interacting with the Open Source Community.**

Implementation Principles

➤ Many principles of system implementation that are relevant to all development are the same now as they have always been at successful software houses.

The steps to implement process and tools in an organization.

Step 1: Assess Development Organization. ...

Step 2: **Plan** Process Implementation. ...

Step 3: Execute Process Implementation. ...

Step 4: Evaluate Process Implementation Effort.

1. Resource Trade-offs

➤ **Resource trade-off** models are spatial models that use biological, environmental and socio-economic data to optimise management (protected area designation) across potentially conflicting uses, or across different ecosystem services.

2 .Frequent Releases

- Recent practice has moved toward very frequent build and release of code, even daily.
- In open source development each build of new code is generally available via CVS, with more stable releases tagged.
- In this scenario, effective testing is made possible by abundant hardware and automated scripts. A product that has been built and tested is a product whose status we understand in full.

3. Support Elements

- In order to control team-based application design and development, we must put a structure in place. All design work and code must be managed through source control.
- Application components must be put in a place where they can be reviewed and reused. Code must be consolidated onto a central server for integration testing.
- In order to tie the pieces together early and keep them working together, application components must be packaged and deployed to staging and production servers.
- Code in all application tiers should be able to run in a debug mode, reporting everything we may need to know. We need facilities to run the application; monitor processing, performance, and results; and compare these to previous runs.

4 .Watching for Problems

- Projects that are running into trouble can be measured technically, such as through bug reports or performance measures. They also reveal themselves in human factors.
- In team communications, there may be an increasing emphasis on secrecy or blame. Items may be piling up waiting for someone else, perhaps a new hire, to do them.
- The schedules of our team members may be expanding, leaving no time for other life activities such as shopping, so those get done during the day.

Key Documents

A **KEY** file is a presentation created by Apple Keynote, a macOS (OS X) and iOS program used to create presentations. It contains slides that may include text, images, shapes, charts, tables, transitions, audio, video, and notes.

Important documents include the project definition, functional, and technical specifications (specs).

1. Project Definition

Every project should have a definition. This begins with the vision phase and is carried through and updated at every phase, and finally evaluated at project completion. Table 4.8 shows the components of a project definition.

Component	Description
Vision	<p>Brief description of project and background</p> <p>Intended users and major scenarios</p> <p>Important goals, whether business or technical</p>
Scope and constraints	<p>Statement of boundaries and areas of impact, including time deadlines</p> <p>Constraints on technology choices, such as standards or interoperability</p>
Risk	Prioritized dynamic list of risks
Costs and benefits	<p>Hardware, software, manpower, whether one-time or continuing;</p> <p>expected returns</p>
Schedule	Project plan with milestones
Team Structure	<p>Full-time team members; virtual team members including partners and users (includes assumptions, skills, critical needs, dependencies)</p>

2. Risk Management

- The risk assessment document is created in the vision phase and then maintained up-to-date throughout the project.
- We will use the risk document to drive risk-driven scheduling. We identify the risky (hard) stuff and attack it first, even if that is not the most convenient approach.
- By attempting to remove risks early, we prepare ourselves for the likelihood that other problems will creep in later.

For each risk, we create:

- Statement that captures the nature of the risk
- Probability that it will occur
- Severity of the problem that would be caused, described and given an impact number
- Mitigation plans that would prevent the risk from occurring
- Contingency plans that would minimize or cope with the problem if it occurs
- Ownership or the party who will monitor the risk

Risk Checklist

The following are all examples of common general risks for information technology projects. Listed first are external risks, which are related to areas outside the control of the project team, followed by internal risks, which are related to the project team and its members and structure.

- Is there sufficient commitment from senior management and users?
- What is the business impact on users, and how much change will they undergo?
- Do users understand the effects and limitations of the technology?
- Are there multiple user branches or sites (or companies or countries) that may have different processes?

3 .Example of a Risk Assessment

Risk assessment is a term used to describe the overall process or method where you: Identify hazards and **risk** factors that have the potential to cause harm (**hazard** identification).

Each is assigned a probability and severity. In this case, they were multiplied to generate a priority by which the table was sorted. Each risk has an owner and mitigation and/or contingency actions.

4. Functional Specification

A functional specification (spec) is a list of features to be included in the project, prioritized and spelled out in sufficient detail. The right level of detail is first a matter of trust and communication.

Outsourced projects often have very detailed specifications. The ideal is just enough detail to enable a developer to build the system that is needed. Too much detail leads to time wasted on a paper process.

5 .Technical Specification

The technical specification should indicate how the functional spec will be implemented. This includes the technical architecture and details of each of the included components. Many people would call this the technical architecture. It should be sufficiently detailed for skilled developers to build the system. It includes:

- Logical model of the system
- Physical model
- Key design points, technology choices, and proofs of concept
- Infrastructure for development, testing, and deployment
- Interoperability, including data flows and events
- Migration, including data conversion
- Security, including authentication/authorization and threat management
- Data stores, including schemata
- Management by operations, including logging and monitoring

Migration

In a migration, risks are higher than with a new system because expectations are higher and users have something to lose. If we have to replace an existing system with a new one, we need to take extra care to plan carefully, test everything thoroughly, and still be prepared for unexpected problems that require backing out changes.

1. Migration Approaches

In existing organizations, there generally is an existing system. There are several different approaches we can take for its disposition. We can:

- Keep the system, recognizing that the system is paid for and works, at least to a point. Abandon the system, outsourcing its function or doing it manually.
- Redeploy the system to another platform with minimal changes—for instance, from UNIX to Linux.
- Develop a replacement system with appropriate new tools and products. This may include redesigning the process more generally to streamline processes.

2 .Assessing the Current System

To determine the right approach, we need to assess both the business and the technology situation. The business problem will be identified in the vision phase and built into the project definition, as described earlier in this chapter.

The business situation has various opportunities and threats, including the actions of customers and competitors and new ideas in business organization.

The areas to assess for a system include:

- ❖ Size, performance, complexity, condition of applications
- ❖ Hardware and software infrastructure
- ❖ Current staffing, service requests, and workload
- ❖ Costs
- ❖ Problems that the current system cannot address

Interacting with the Open Source Community

We will interact with the open source community in several ways. We will have open source developers on staff, and need to consider how to hire and retain them. We will interface with open source products by using them, and may have opportunities to contribute to them with code, work, or finance.

1. Hiring from the Community

A few people in open source are famous in a general sense, but, much more importantly, at the level of code contribution to particular projects, many people have built reputations within a particular community.

Open source is open and public, so you can see code, written postings, and so on that you would never see in a candidate from a closed code company. It may be a good idea to use those resources.

In some cases, you might want to hire the maintainer of a code project if it's important to you.

Martin Fink of Hewlett-Packard cites a “two-hop” rule. If a project is important to his organization, he likes to know that he is two people away from a maintainer or key contributor to the project.

2 .Employee Agreements

There are several issues where organizations generally do not have policies today, but may need to develop one. This may involve a review of relevant employment contracts. The following are some examples, but this is not an exhaustive list; there may be other issues.

Some employees will want to be allowed to work on open source projects while on organization projects—for instance, by sharing utility code or returning enhancements made to open source software.

This is reasonable, but may conflict with current employment agreements. Others may wish to contribute to open source software on their own time. This is also reasonable, but many organizations have blanket policies prohibiting it.

3. Repaying the Community

Organizations that benefit from open source software often develop methods for repaying the community. The simplest can be allowing employees to work and contribute to the community, as well as serving as a reference and otherwise being a good citizen.

There may be opportunities for sponsoring enhancements that are relevant to your organization's use. By directing investment toward enhancements your organization needs, you may gain leverage in the direction of the product.

11. Support

Support costs include software maintenance, internal resources, and contracted external resource. Software maintenance fees, typically annual charges of about 20 percent of the purchase price, are simply a cost of doing business when using many closed code program, which you will escape by using open source software.

These two different requirements interact. Often, reports on problems and missing features in software must be resolved into user or software error. Specifically, we can resolve support into three levels, as follows:

Level 1: Report and identify issue, assign ID, triage and resolve if possible.

Level 2: Get the issue to an expert in the problem area and resolve without development request.

Level 3: A bug is filed, and a patch developed in the field and incorporated or the problem is worked on by a developer.

UNIT – V

Application Architecture: Types of Systems- Tiered Design- Managing Performance and Scalability- Interoperability- Development Platform Choices.
The Cost of Open Source Systems: Total Cost of Ownership- Types of Costs- Scenarios- **Licensing:** Types of Licenses- Licenses in Use- Mixing Open and Closed Code- Dual Licensing- Other Intellectual Property Issues.

Today's Topics to discuss...

Application Architecture:

Types of Systems

Tiered Design

Managing Performance and Scalability

APPLICATION ARCHITECTURE

In this chapter, we will look at different classes of applications, and then review some key design points for applications that will perform well.

An **application architecture** is a map of how an organization's software **applications** are assembled as part of its overarching enterprise **architecture** and how those **applications** interact with each other to meet business or user requirements.

Then we will cover the methods for loosely coupled communication with other systems, which can include mainframes and Java application servers.

Finally, we review the development platforms available and position the open source choices, which are LAMP and Tomcat/JBoss, against the major closed code products, which are server pages and application servers based on .Net and Java.

The term LAMP refers to development with Linux, Apache, My SQL, and PHP. Less commonly, it might include the other open source languages Perl or Python or the open source database Postgre SQL.

1.Types of Systems

If an organization is planning to implement a complex enterprise application, the possibility of running it on an open source platform should be reviewed.

At this time, it is usually possible to run the major enterprise applications, such as those from SAP, Oracle, and PeopleSoft, on Linux servers.

There are fewer applications available that are open source from top to bottom, but there are some available and more can be built.

This type of deployment has only been happening recently, yet already there are over 2,000 SAP deployments and over 500 Oracle deployments on Linux.

In either case, it is now time to plan in order to deploy in that time frame.

Since Apache is the most popular Web server and PHP the most popular Web development tool, it is not necessary to demonstrate that the majority of normal Web applications can be built this way.

A cursory search of the Web reveals large numbers of attractive and reliable applications that have been built using the open source tools (LAMP) and that there is a large body of people available who can develop and maintain these applications.

Extreme Systems

The development tools discussed in this book will handle many complex distributed applications.

However, many of the largest and best-known systems are really extreme cases that employ exceptional methods and are not typical of the way other business systems will be built.

They are often the first application of their category, the largest company in their industry, or a unique organization such as a clearing-house. Examples include:

- ❖ Travel reservation systems, such as Sabre and Amadeus
- ❖ Banking clearing-house systems, such as SWIFT
- ❖ Exchange systems, such as the New York Stock Exchange (NYSE)

- ❖ Large organizations, such as Wal-Mart, Bank of America, and Citicorp
- ❖ Leading dot-com companies, such as Google and Amazon.com
- ❖ These organizations have developed substantial innovative software systems using specialized transaction processors and low-level languages.
- ❖ Sabre and NYSE use Tandem (now HP NonStop) systems. Bank of America and Sabre use the custom mainframe IBM Transaction Processing Facility.
- ❖ The language used in these systems is C++ or older low-level languages, including PL/I and assembler.
- ❖ This behavior is really more like software product developers than typical business systems.

When applications need the highest performance, the language to use is always C (or C++).

For example, Microsoft generally finds Web applications to be about twice as fast running benchmarks when written in (unmanaged) Visual C++ instead of VB and ASP.

Other vendors just don't do database benchmarks except with C++, because their application servers are even slower.

C++ is used with ISAPI rather than ASP, direct database access rather than, memory caching rather than simple database access, and so on.

Transactional Systems

Transactional systems are databases that record a company's daily transactions.

Normal development in organizations is not the same as software product development or these extreme systems.

Leaving aside these special cases, almost all transactional Web applications can be constructed with “server page technology,” such as PHP, ASP, or JSP, with some sensible use of server-side components.

The following optimization techniques, which are described later, can give order of magnitude improvements in performance but must be designed into the application:

- ❖ Effective state management
- ❖ Queuing
- ❖ Good database design

Knowledge Management

Not all applications are transactional. Other applications typically have fewer users and offer a less directed and predictable set of interactions.

This includes many kinds of information and support systems and forms of collaboration; we will put them all in the category of knowledge management.

These applications usually interoperate with transactional systems indirectly.

Knowledge management is the conscious process of defining, structuring, retaining and sharing the **knowledge** and experience of employees within an organization.

The main goal of **knowledge management** is to improve an organization's efficiency and save **knowledge** within the company.

Knowledge management applications also commonly interact with infrastructure systems.

They typically rely on the mail system to deliver messages, the directory for authentication, or the file system as a data store for documents.

They commonly need programmatic access to these system interfaces.

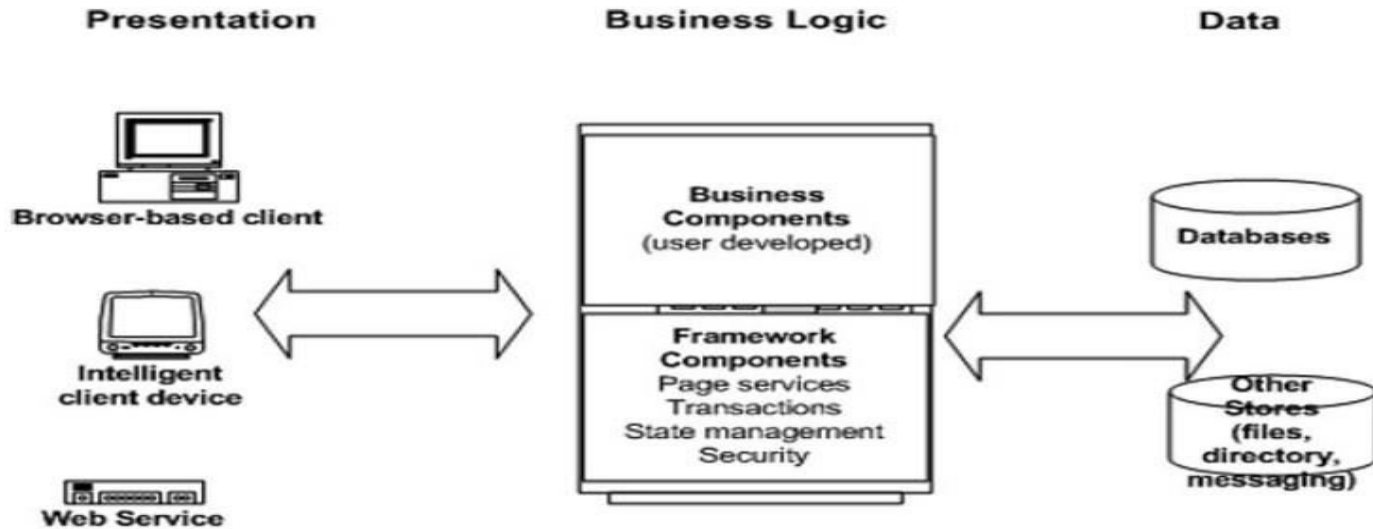
Transactional systems typically will not want to share resources with such systems, because they need to manage their scalability separately from the general organization.

2 .Tiered Design

We will design applications as a number of logical components. This breaks down complex problems into smaller pieces and hides implementation details from calling components.

Components for business applications generally have recognizable tiers for user interface, business rules, and data, although this does not have to be a hard and fast rule.

In software engineering, multitier architecture or multilayered architecture is a client–server architecture in which presentation, application processing and data management functions are physically separated.

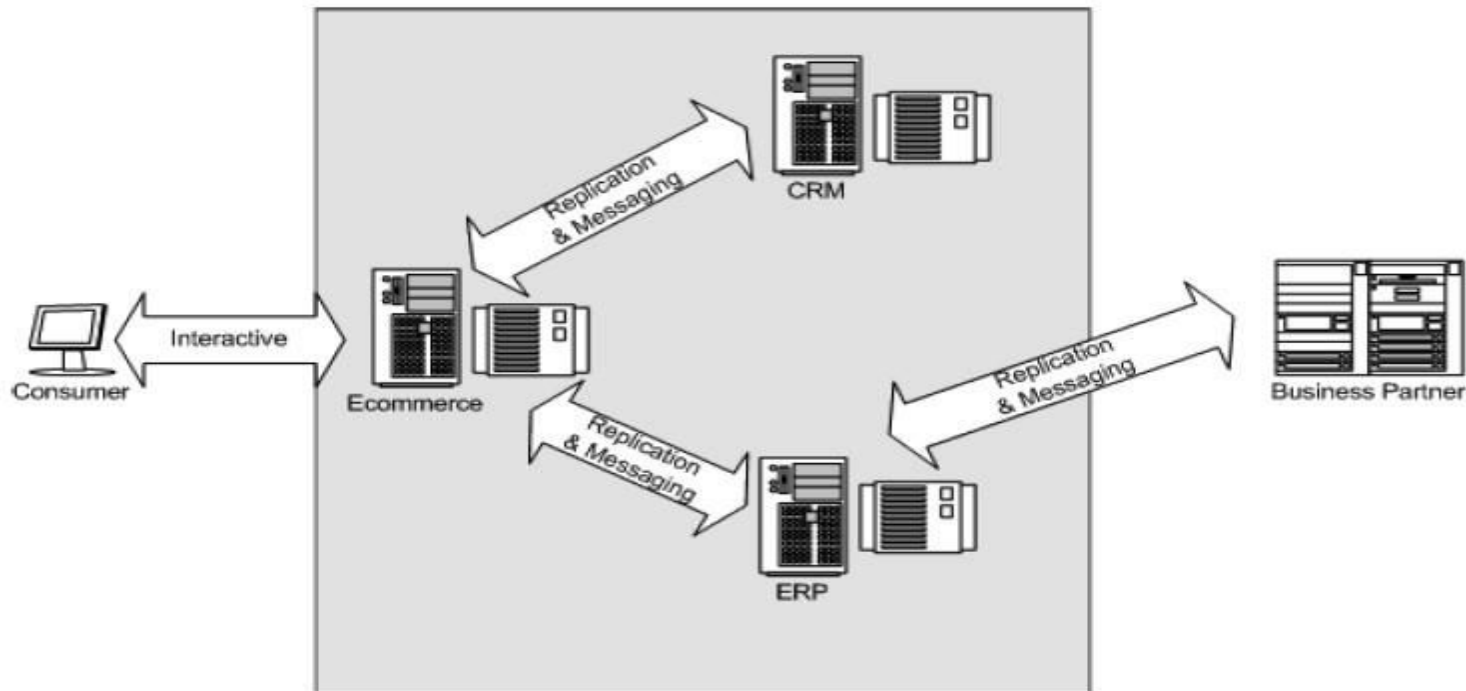


In fully connected networks, business rules usually go into server-side components, to allow reuse and to avoid deployment problems. The trade-off here is possible duplication of code connected with validation.

3. Managing Performance and Scalability

What we want to achieve is loose coupling on commodity platforms. This provides scalability and is also best for costs, so the result is the best price / performance.

Loose coupling is an approach to interconnecting the components in a system or network so that those components, also called elements.

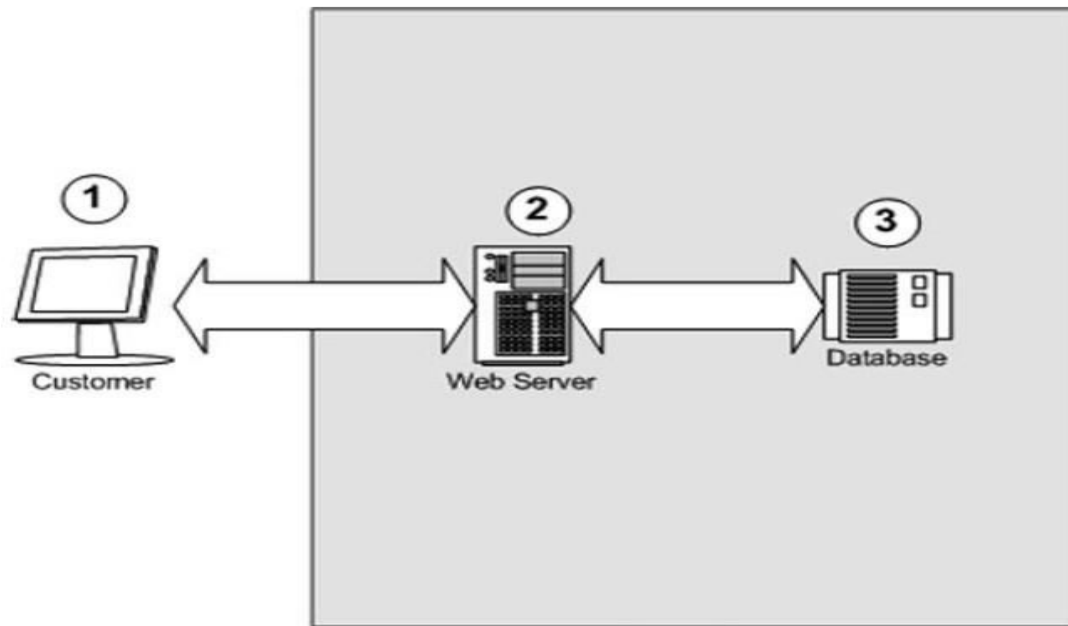


Enterprise resource planning (ERP) refers to a type of software that organizations use to manage day-to-day business activities such as accounting, procurement, project management, risk management and compliance, and supply chain operations.

Customer relationship management is one of many different approaches that allow a company to manage and analyse its own interactions with its past, current and potential customers.

1.State Management

State management refers to the **management** of the **state** of one or more user interface controls such as text fields, OK buttons, radio buttons, etc. in a graphical user interface.



We can maintain transient state at the Web server, and we will get good performance in the sense that we cut down on network traffic.

However, we lose a great deal of function by giving up the database. We do not get transactions, which can give us a lot of extra work in the application or may expose us to corrupt data after a Web server failure.

This model also limits our ability to use distributed systems;

Finally, by holding server resources during a long-running transaction, we use them over much longer time frames because user time is orders of magnitude larger than machine time.

We can maintain transient data in the database, either in regular database tables or special formats, such as temporary tables or dedicated dictionaries as in commerce products.

If we manage state at the client, we will scale well, but we face issues of management, security (e.g., tampering with prices), privacy (cookies), and client identification (AOL caching issues).

2. Queuing

computing to arrange (a number of programs) in a predetermined order for accessing by a computer.

One of the most important elements of design for performance is to use asynchronous methods correctly to decouple components from each other—for instance, front end from back end.

There are several names for this but I'll simply call it queuing. A correct use of queuing was behind many of the most successful transaction processing systems of the last 40 years.

Queuing offers improvement to an application in reliability and throughput. The first advantage of queuing is that it increases reliability.

Queuing decouples a business transaction from issues of server, client, or network availability—in other words, it can provide “success in the face of failure.”

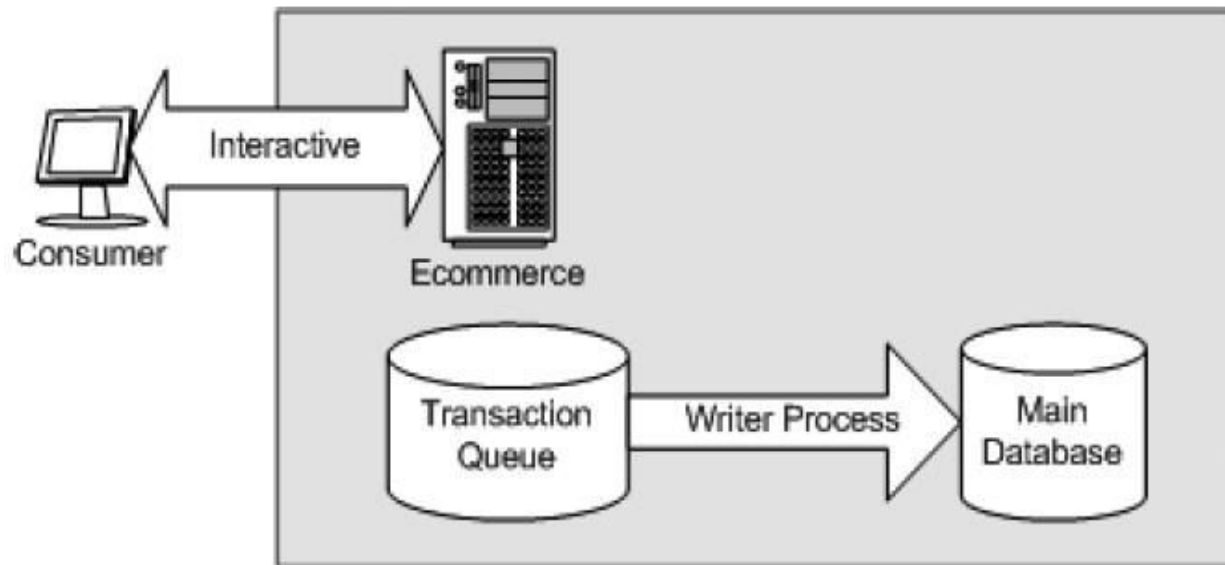
For example, our order processing may depend on a shipping component. But possibly shipping faces delays or is not in operation all the time. We can acknowledge orders now and then queue them for later processing and shipping.

The second advantage of queuing is that it allows us to economically manage large variations in demand.

Businesses do not experience consistent demand for services, but have many cycles from hourly and daily to periodic, and also may experience heavy volumes caused by unusual events.

Queuing is not for every application, because of the extra work it introduces. This includes:

- ❖ Marshalling the data into and out of the queue format
- ❖ Handling business error situations, which can occur if, for instance, an order is not fulfilled
- ❖ Replication of data, such as catalogs



Queuing in the database.

Queuing is a very good method for loosely coupled communication between applications.

One obvious way to get queuing is to employ a “message queuing” product.

While messaging can be implemented over a database, mail, HTTP POST, or FTP, message queue products can provide higher-level facilities including serialization, and support for long-running transactions and transaction reversal.

There are several open source tools for messaging, including JBoss, and several Java Messaging Services (JMS) products, OSMQ and MOM4J are two open source products that implement JMS for open source languages in addition to Java.

3 .Database Design

For large systems, we will need to optimize the database.

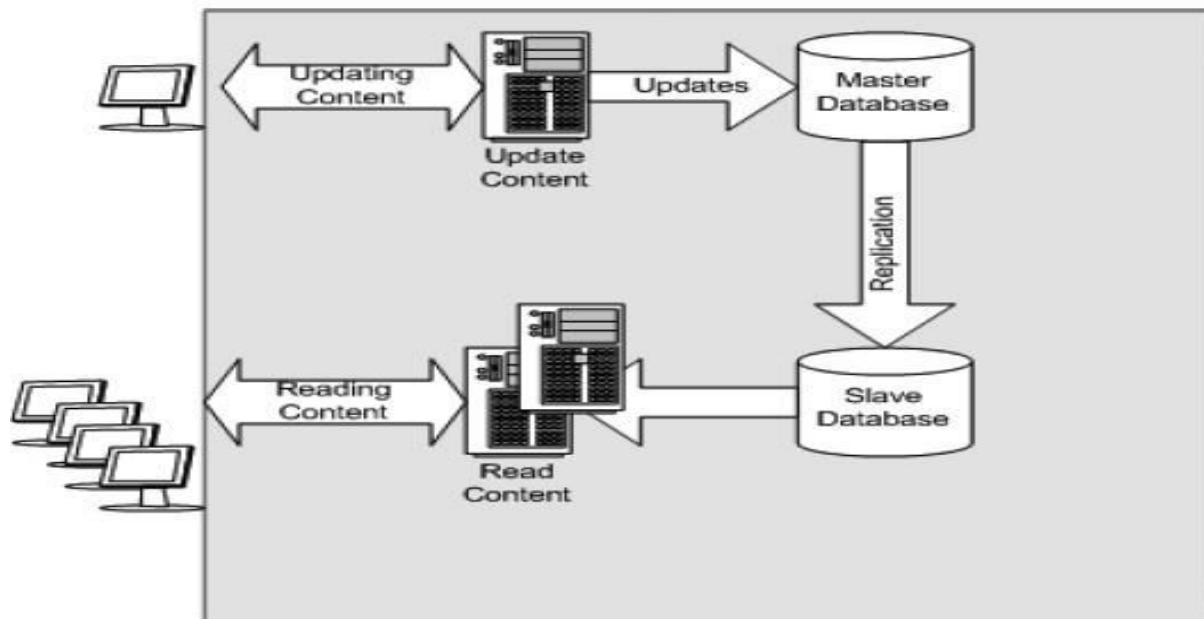
I have spent several years in a consulting practice reviewing databases; our team was usually called in when the customer had decided to replace the database with another because of lack of performance.

Our experience was that we could always get much more performance, often as much as an order of magnitude.

The techniques to do this are as follows:

- Measure performance, isolate the problem areas to allow focus on the important areas, and use tools to see the actual database queries running.
- Review the application requirements and design against the database design. It is common for implementers of the application not to be aware of data volumes, and database administrators not to be familiar with application logic.

- Use database-specific optimizations, such as stored procedures, cursors, indexes, and prepared statements, correctly. In MySQL, this includes choosing the storage type.
- Find a clean way to pool database connections so that the pool is maintained, rather than continually connecting and dropping.
- Replication can be used to separate update transactions from read-only.

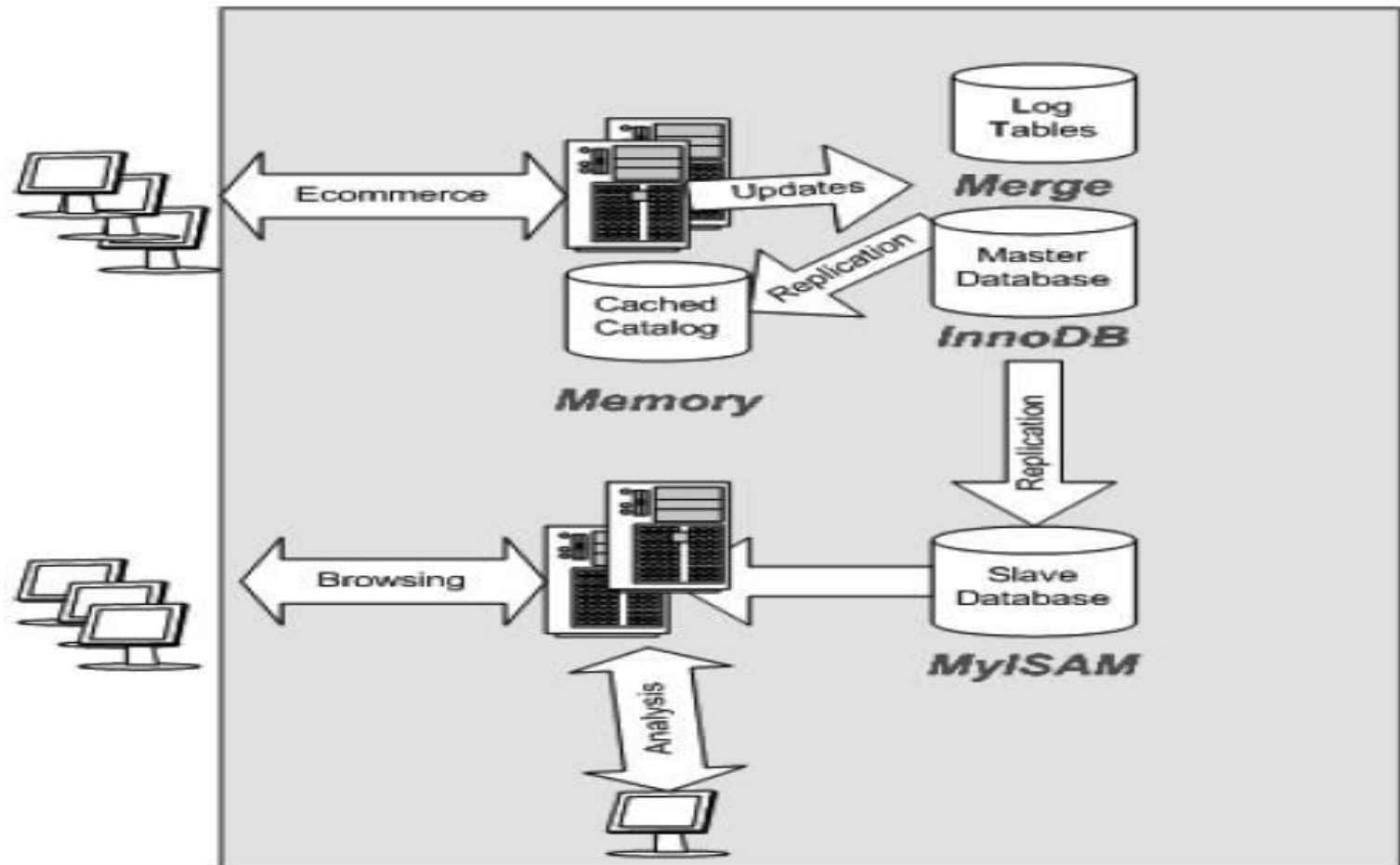


Storage Engines

MySQL offers several table types, which refer to different storage engines. The two most important are InnoDB and MyISAM. The MyISAM type has no transactions and is not good at handling mixed reads and writes, but it is fast at writing or reading.

Two other types are useful for specific purposes. The Memory type is fast but has no disk storage, so is appropriate for caching. We can copy translation tables, pricing, and other nonvolatile information into a cache to avoid database access.

The Merge type can be useful for logs, since it allows us to maintain a different physical file for each day or week of activity, yet combine them when needed.



Combining MySQL storage engines

4. Application Servers

Application servers provide a number of useful things. They provide transactions including transaction composition, automatic failure handling, and two-phase commit.

They allow us to split our application into parts, in some cases across languages and particularly across system boundaries.

They perform the work of sharing some number of users across an arbitrary number of processes.

They give us thread and process management in a language-independent manner while hiding the details.

They manage resource pooling, such as database connections.

Previous Topics:

- ❖ **Application Architecture:**

- ❖ Types of Systems

- ❖ Tiered Design

- ❖ Managing Performance and Scalability

Today's Topics to discuss.....

- Interoperability
- Development Platform Choices
- **The Cost of Open Source Systems:**
- Total Cost of Ownership

Interoperability

Interoperability is a characteristic of a product or system, whose interfaces are completely understood, to work with other products or systems, at present or in the future, in either implementation or access, without any restrictions.

A typical large organization has several distinct environments. Applications written to run on any of these platforms must be able to share information and interoperate with any of the others.

So it is important to have a plan for interoperability between systems running different software on those different operating systems.

These include:

- ❖ Microsoft Windows infrastructure with COM and/or .Net development
- ❖ Systems based on Java application servers, such as generally on UNIX or Linux
- ❖ Linux systems with LAMP applications

Strategies for interoperability can be placed under the broad headings of shared access to data, including file transfer and shared database, and process communication, including Web services.

Whichever approach is chosen, if data formats are complex, as they generally are, there is a need for data sharing standards.

These are usually structured around XML. The two preferred alternatives are shared database and Web services.

1 .Shared Data

We can share data using a file system or a database. For knowledge management applications, we will often use file transfer or shared access to files.

This can include FTP; network file sharing with Samba, NFS, or iFolders; or files sent as mail attachments.

This is a good way to manage slowly changing data that is distributed from a central point. It can also be used to replicate databases in some circumstances. We can map a database into XML and send it as a file.

here are some basic problems of file sharing that must be addressed. Windows, Linux, and the Mac encode text files differently, and binary data is subject to platform variations. Network file shares are subject to more frequent and different types of error than local files.

2.Process Communication

Process communication generally involves development. It includes:

TCP/IP socket programming

Screen scraping

Messaging

- Application integration engines

- Web services

Socket programming is a good mechanism for tightly coupled communication.

A **socket** is one endpoint of a two-way communication link between two programs running on the network.

It is not always as easy as one would wish, and there are data marshaling issues, but use of a high-level language such as Python and XML for encoding makes this reasonably simple. Network errors are difficult to handle.

3 .Application Integration Engines

Application integration is the process of enabling independently designed **applications** to work together.

The general-purpose integration engines such as Web Methods, BizTalk, and IBM Web Sphere Integration Server can sound appealing.

Integration engines offer an attractive interface, which demos well, and a long list of packages they can work with.

These tools are sold to by pass developers by using pretty GUIs and such, but they don't by pass developers (because the details really are complex).

These tools are expensive, so there is often pressure to use them in several projects to amortize the cost.

There may not always be a good fit. In practice, most people write their own solutions for application integration.

At any time, most people have a point-to-point integration need, which can usually be addressed with a program in a scripting language such as Python.

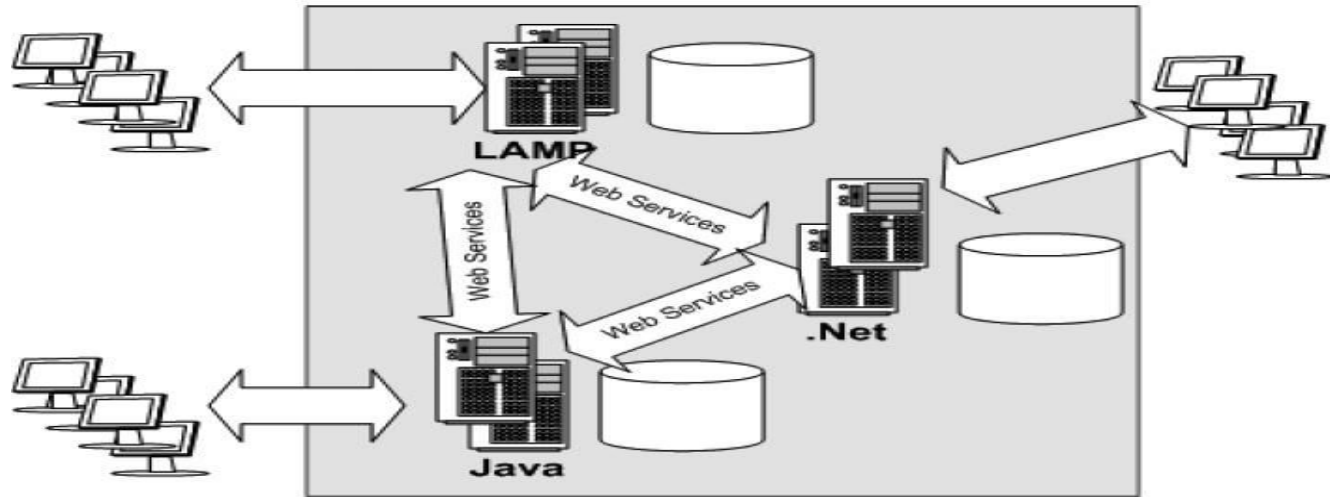
4. Web Services

Web services offer a standard method of communication that is independent of language and platform. Using Web services, we can make a call to another system regardless of the platform on which it runs.

The data is marshaled and tagged in XML, and the call is made across platform and operating system.

we can use Web services to achieve interoperability between WebSphere, .Net, and LAMP applications.

This can be achieved at quite small levels of granularity, such as a function call, or large levels, such as a file of purchase orders.



Web services can be asynchronous or synchronous. Asynchronous calls allow for loosely coupled systems, particularly combined with document style XML to allow transfer of large quantities of related data.

The simplest form of Web service is XML-RPC. XML-RPC may be the best choice for internal use and rapid development. SOAP is a more elaborate standard, with more features and choices. SOAP is more likely to be used for industry standards or inter business communication. All of the open source languages have good support for Web services.

5 .Data Formats

The problem of data marshaling, which is an issue for most of the interoperability techniques, occurs at several levels.

At the next level, we need a system for tagging field names, then for data typing, and then to manage linkages between fields to support hierarchical representations.

Above this are semantic issues, such as agreement on the content and relationships of fields. Shared database solves many of these directly, but more complex relationships still need to be mapped to XML.

Development Platform Choices

For corporate development today, the choice generally comes down to one of the following:

- Java, including Java Server Pages (JSP) and/or Java application servers such as WebSphere, WebLogic, and Jboss
- Microsoft .Net, including C# or Visual Basic with ASP and the CLR run-time environment
- LAMP (Linux-Apache-MySQL-PHP)

Any of these approaches can be used for rapid development of simple applications and then scaled up where necessary, at some trouble and cost, to produce large architected solutions that meet complex requirements with high performance.

All of these can be developed to run on various servers, including Windows and Linux.

All are cross-platform at the client if developed as Web applications and tested correctly, and all can create client / server cross-platform code under the right circumstances. We'll compare them in some detail.

1. Java

We can build simple applications with JSP (Java Server Pages), which is how most Java applications are done.

We can build complex applications with a Java application server using a framework such as Struts.

The leading open source choice for Java Server Pages is Apache Tomcat.

Application servers include JBoss, the leading open source application server, or one of the closed code application servers:

- ❖ IBM WebSphere
- ❖ BEA WebLogic
- ❖ Oracle Application Server
- ❖ SUN ONE

JSP and Java applications are cross-platform across Linux, Windows, UNIX, and other servers, as well as across clients, including Windows, Mac, Linux, and others, if developed and tested correctly.

Pure Java Server Page applications are pretty portable. Unfortunately, applications that use the vendor application server environments such as WebLogic and WebSphere are difficult to move between vendors, since these environments are integrated with proprietary vendor tools and services.

The main strength of the Java approach is its cross platform, which is excellent. These systems can be scaled for high performance, albeit at a high price.

The weaknesses of the Java approach are cost and difficulty of development. While the majority of Java systems have been developed using JSP, and probably could have been done with open source approaches, most organizations have used expensive tools to do it.

2 .Net

With this approach, we can build simple applications with ASP.Net using C# or VB. We can build complex applications using tiers, with COM+ as the application server and SQL stored procedures.

Many dot-net development tools are available from Microsoft and third party tool vendors for the Microsoft platform; of course, most developers will use Visual Studio.

Dot-net is noticeably less expensive than a full Enterprise Java solution. In fact, ASP.Net is arguably free with Windows 2003 Server if you can forgo the Visual Studio development environment and use a programming editor instead.

3 .LAMP

Using this approach, we can build simple applications with PHP, Perl, or Python.

We can build complex applications in the traditional way for open source software, tiered with a scripting language in front and C/C++ server and library elements as necessary, or we can use the JBoss application server.

All of these systems are cross-platform at the client if developed as Web applications and tested correctly.

LAMP applications are cross-platform across Linux, Windows, UNIX, and other servers, as well as across clients, including Windows, Mac, Linux, and others, if developed and tested correctly.

The strengths of the LAMP approach are cost, cross-platform capability, and ease of development. The potential weakness of the LAMP approach is complexity growing with scale.

Successful LAMP implementations include many small to medium systems, which have been developed simply, and some very large ones, including Amazon, Sabre, and Slashdot.

As discussed earlier, some very large applications will need to have some seriously competent C++ programmers available.

THE COST OF OPEN SOURCE SYSTEMS

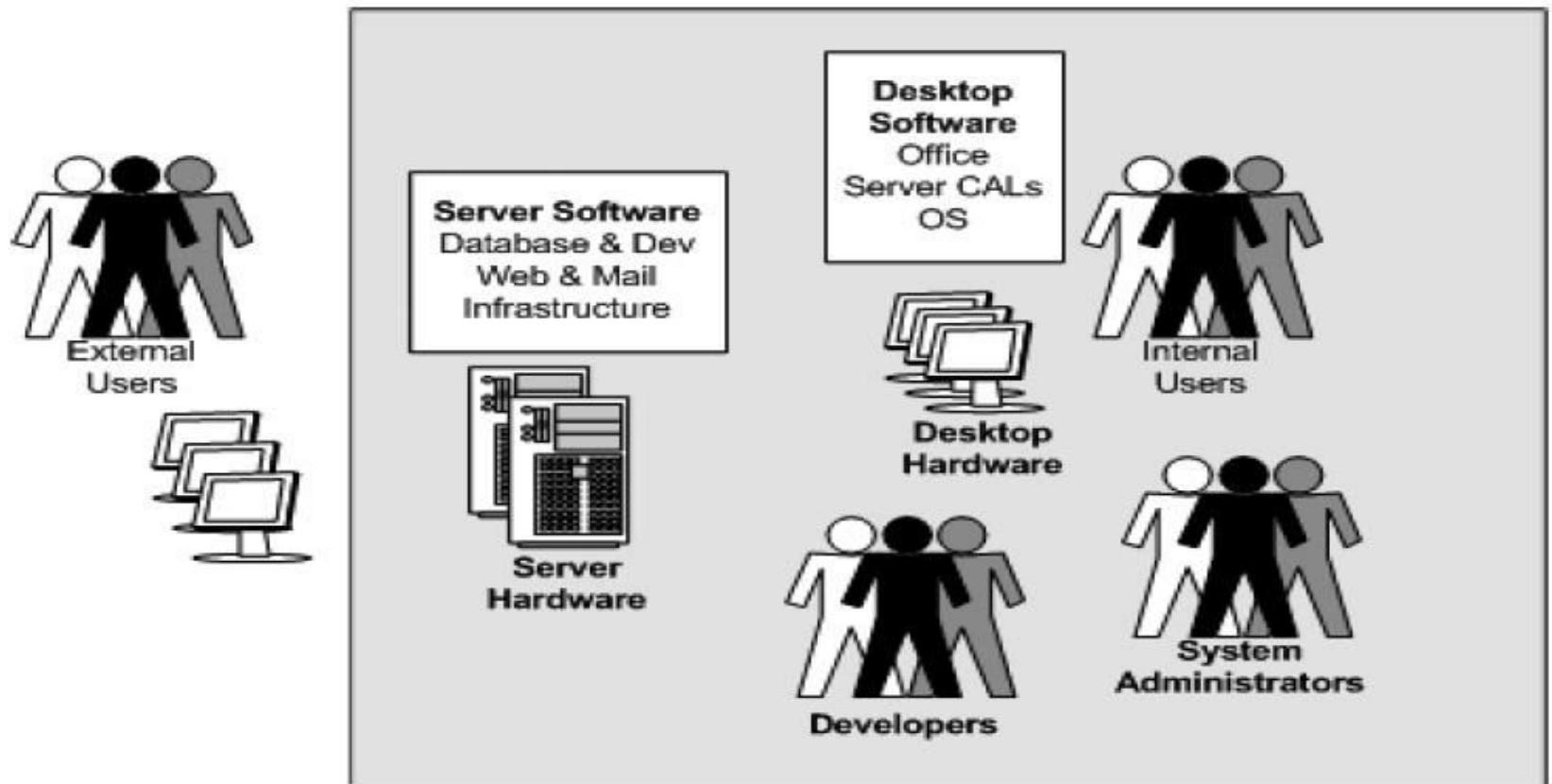
Of course, it is not difficult to price open source software products, since they are almost all free.

The difficulty comes when we move past this to do these more difficult things:

- Price comparable closed code products.
- Price all associated costs, including hardware, staffing, and support costs, to derive a total cost of ownership (TCO).

we will compare open source software prices with similar closed code software. Of course, there are not always similar products and where there are, we may have a preference for one feature set over another. In this chapter, we will only compare costs.

Then we will examine the total cost of ownership of open source and closed code products and compare those.



Cost elements

Total Cost of Ownership

There is a simple answer to the question of open source software costs, where open source solutions are comparable to closed code alternatives.

When compared with similar closed code systems, open source systems as a general rule cost:

- Much less for software
- No more and often less for hardware
- If other things are equal, no more for anything else

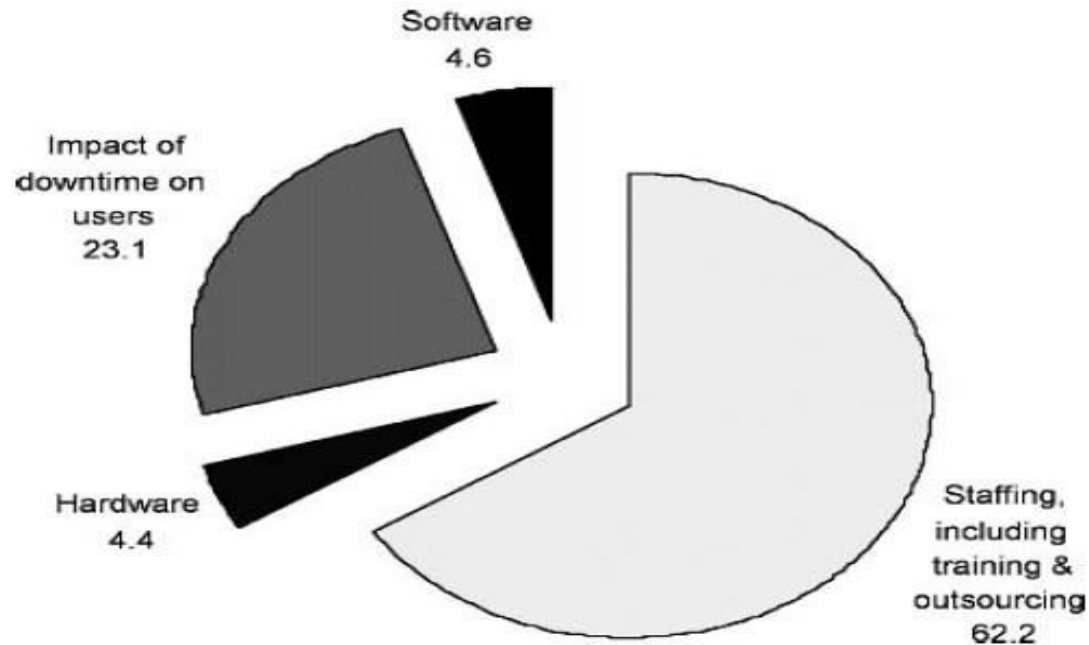
As far as software costs are concerned, we will review tables with the prices for common open source and closed code products, and see that open source software costs much less.

As far as hardware is concerned, open source products are available for effectively all current hardware platforms, including the systems with the best price/performance.

Open source performance on a platform is usually similar to closed code competitors, as already discussed throughout this book.

1.Staffing Costs

Personnel costs dominate software costs for infrastructure. Because of this, the savings from open source software such as Linux and MySQL will be small compared with the costs of personnel for development and management.



This may understate software costs, but it is broadly consistent with work by Gartner on IT costs, which again shows staffing and downtime as the major costs for infrastructure.

Application solutions can be much more expensive. Large applications can incur millions of dollars in costs for software acquisition or development, as well as large server hardware costs, particularly for database systems.

Even for simple Web applications, hardware and software are higher than for infrastructure.

The costs are loaded, including salary, vacation, management overhead, general training, taxes, and benefits. They are averaged, with no effort to distinguish between skill levels.

There is also an entry representing a week of training. Many projects require training of a week or two for developers and administrators.

2 .Hardware Costs

Hardware costs include servers, clients, networking equipment, and other appliances such as firewalls.

Hardware costs are generally about the same between Windows and Linux, unless there is some unusual performance issue causing a difference. Usually, the same hardware can be used at all levels.

3.Software Costs

The closed code comparative software prices that follow are given for Microsoft. The major vendors track each other's pricing and Microsoft's pricing is more transparent than other larger vendors.

In my experience, Microsoft is very rarely more expensive for the same class of product than IBM or Oracle, and its pricing is relatively stable, easy to get, and easy to work with.

Previous Topics:

Interoperability

Development Platform Choices

The Cost of Open Source Systems:

Total Cost of Ownership

Today's Topics to discuss.....

- Total Cost of Ownership
- Types of Costs
- Scenarios
- **Licensing:**
- Types of Licenses
- Licenses in Use
- Mixing Open and Closed Code
- Dual Licensing-
- Other Intellectual Property Issues.

Total Cost of Ownership

4 .Using Third-Party Application and Database Servers

In both the open source and Windows environments, there are many alternative choices of third-party tools and database servers.

Popular choices include:

- Oracle or IBM DB2 database servers
- IBM Web Sphere, BEA Web Logic, or Oracle application servers
- Tools for modeling, debugging, code management, and so on, such as Rational and Clear Case

These products have the same performance and functionality and are about the same price in either environment.

People who choose these products generally choose them at least partly for this ability to offer the same experience across the Windows and Linux platforms; they do not see the Windows-only tools as equivalents.

5 .Pricing Open Source Software

Note that you can always distribute an open source software product, so you only need to buy a single copy to get documentation and CDs.

The two exceptions here are My SQL, which is sold under a commercial license priced per server, and Red Hat Enterprise Linux, which is only sold including support, so that is also priced per server.

6 .Pricing Closed Code Software

It is difficult to fully determine closed code software costs for several reasons. Not all systems have a published price list, and the lists that exist are incomplete.

Products are often offered with very different prices to different customers, and even different pricing models.

Most companies offer substantial discounts, which are not published, to large customers. Some products are only available through personal contact and quotation from a salesperson.

7 .Pricing Windows Software

Microsoft has a published price list, so we can work with those numbers.

It is often a good practice to compare list prices, since discounts are unpublished and can vary considerably.

Although list price comparison usually tends to be roughly fair, it is not fair when comparing open source with closed code.

Closed code software has higher prices and is often discounted considerably, so ignoring discounts will tend to over count the price of the closed code.

Types of Costs

We must take into account several cost factors that weight heavily, including fixed, off-budget, sunk, and switching costs.

In a direct comparison of two new systems, where things are equal, open source software will be less expensive in almost every case.

But often the comparison is in some sense a migration, where there will be a big advantage to the most likely Windows today.

Decision makers may be unfamiliar with open source and inclined to assign higher risks or expect to pay more for services.

1 .Fixed Costs

A **fixed cost** is a **cost** that does not change with an increase or decrease in the amount of goods or services produced or sold.

Fixed costs are **expenses** that have to be paid by a company, independent of any specific business activities.

Examples of fixed costs include rental lease payments, salaries, insurance, property taxes, interest **expenses**, depreciation, and potentially **some** utilities.

If we are funding a project that brings technology to a new population, we will have to consider these costs in the project, but they will in any case be the same for open source or closed code.

2.Off-Budget Costs

This refers to expenditure that's not funded through the **budget**. ... So, even though the borrowing and spending is **outside** the **budget**, it has implications for the **budget** and for all practical reasons should be included in that document.

The effect on users may sometimes be reflected in penalties related to a service-level agreement, but more commonly as a constraint on the IT organization, which must maintain a particular level of service.

The effect of off-budget costs, when included, is to make estimates of user downtime and dissatisfaction the largest elements of the cost models, although these are very difficult to measure objectively.

3 .Sunk Costs

Sunk costs are the costs already spent on existing systems and are not recoverable. It is difficult to get money for old systems, particularly after the dot com bubble;

4 .Switching Costs

Switching costs are the additional costs it will take to move from an existing system to a proposed new one, as opposed to keeping the existing one.

The effect of switching costs is to make new technology harder to adopt.

The first application with a new technology will cost more than subsequent ones, because of training of developers and administrators, who presumably know the old technology, and because of first purchase of servers, development tools, and other infrastructure that will be reused for future applications.

Scenarios

The following examples illustrate two companies with greatly varying numbers of internal desktops.

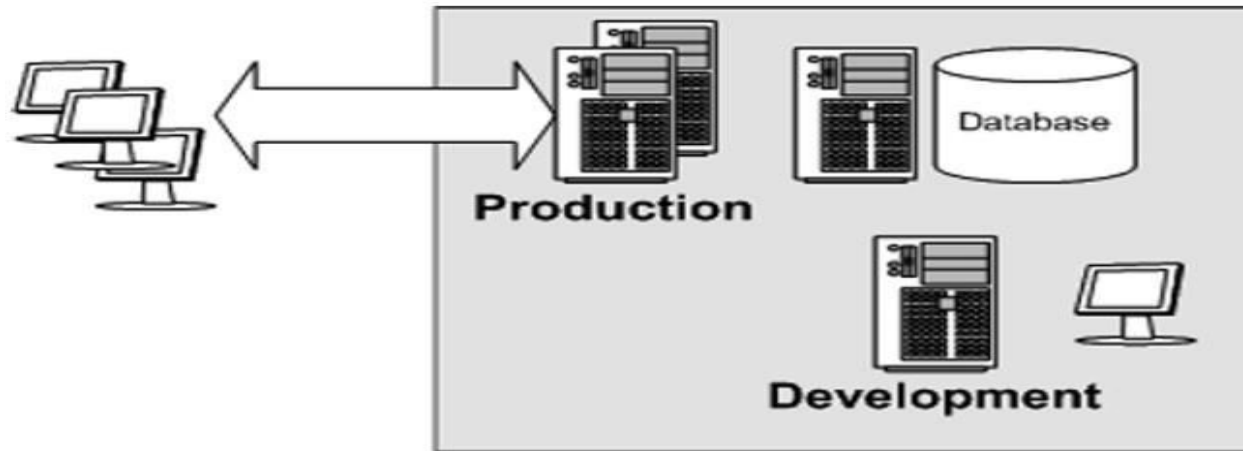
They have different scenarios of internal and external applications that represent different issues.

We will calculate likely costs for infrastructure, development, and application products separately and then sum them.

Costs will include hardware (expected to last three years), software purchase, software maintenance (typically 25 percent of the purchase cost per year), development (one-time), management (for the three years), and any switching costs (one-time).

1 .Small Organization: Web Site

This is a small company with an ecommerce server. The system has a small production database, a pair of Web/application servers, and a development box, all of which are small two-processor servers.



Small ecommerce system.

The system uses four two-processor servers: two Web servers, a database server, and one system for development and testing.

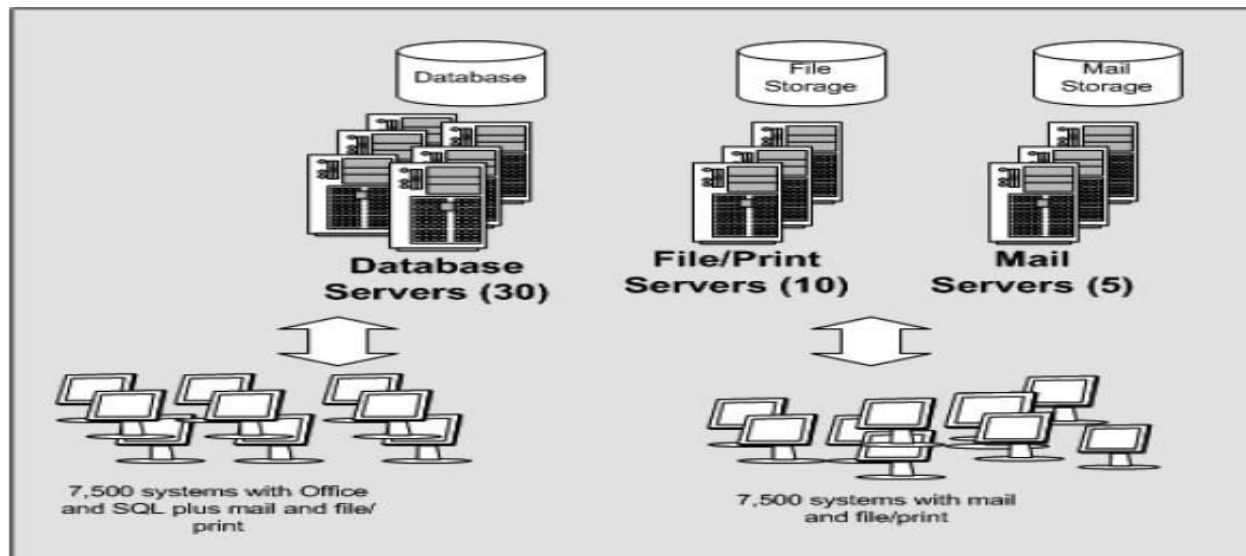
System administration would be a quarter of a resource annually, priced over three years. Databases are priced with the unlimited user license, which is usually the best for Web systems.

The Linux system with open source software is less expensive than the Windows one. If you add Oracle and an application server, you get the results shown in the third column, where the Linux system is more expensive due to the cost of these two items.

2. Large Organization: Internal Use

In this case, we assume that there is a custom application using a relational database supporting 7,500 desktops. There are 7,500 users running Office; from a cost point of view it does not matter if this is the same or another group. All users are supported with a mail server (Exchange in the Microsoft case) and file and print services.

A large company such as this would probably save over \$2 million through Select or Enterprise licensing.



LICENSING

Software licensing has always been a part of the process of managing systems. The issues around open source licensing are not really different from licensing in general, but they do seem to receive more attention at the moment.

Many professionals find legal issues and, in particular, licensing, one of their least favorite parts of the job. However, it is essential for all of us to know the basics of licensing. We will cover the basics in a simple way here. If our needs are more complex, we will require a lawyer.

Types of Licenses

Open source licenses can be divided into two groups: the reciprocal or “free” licenses, of which the GNU General Public License (GPL) is best known, and the nonreciprocal or “open” licenses, such as the BSD and Apache licenses.

1.Relicensing Only Matters If You Distribute

Some people use the term *viral* for reciprocal. The implication is that handling viral licenses is dangerous, as Microsoft sometimes suggests. It is true that Microsoft needs to be careful using products licensed with the GPL.

2 .Reciprocal Licenses Are Similar to Commercial Licenses

Reciprocal licenses are quite similar to commercial (closed source) licenses, which commonly contain terms that restrict relicensing and distribution of information.

A common commercial restriction prevents us from relicensing the software or derivative works.

1 .Reciprocal Licenses

Reciprocal license means a **license** of an item of Software that requires or that conditions any rights granted in such **license** upon: (A) the disclosure(facts or information known to the public), distribution or **licensing** of any other Software (other than such item of Software in its unmodified form), (B) a requirement that any disclosure.

2 .Nonreciprocal Licenses

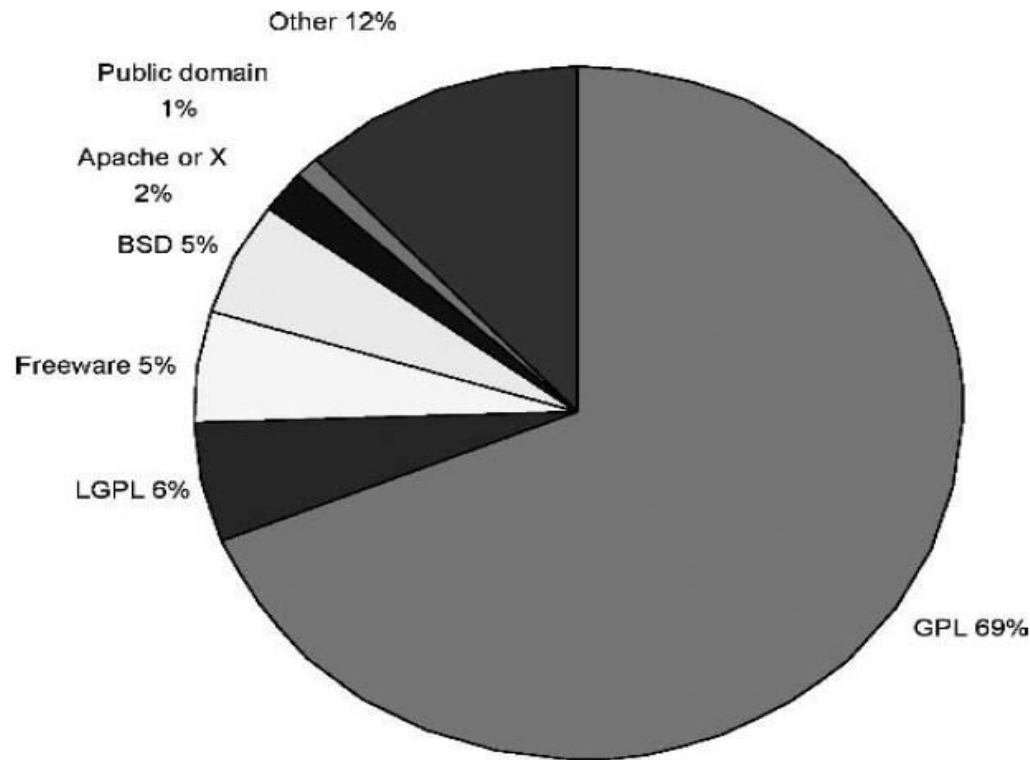
The other licenses (LGPL, BSD, Apache, and MIT) are nonreciprocal. The Lesser General Public License (LGPL) is a nonreciprocal version of the GPL intended for certain libraries. There are two forms of the BSD license.

The nonreciprocal licenses are less restrictive than the GPL on distributors. Subsequent users can use, modify, and redistribute the code without distributing their source code.

This lack of restriction for the distributor removes the rights of users downstream from them to see that code.

Licenses in Use

There are many licenses in use today, but only a few that need to be considered by most organizations. The Fresh meat site lists about 50 categories of licenses, some of which are groups of licenses, but only about 20 are used by at least 100 projects.



3 .Which License to Use

It is strongly recommended that if you are distributing your own open source product you adopt one of these licenses without alteration:

- GNU General Public License
- Mozilla Public License
- BSD, Apache, or MIT license
- GNU Lesser General Public License

Mixing Open and Closed Code

It is quite possible to use closed code and some open source software together. This is common today and is likely to be the way most systems are built in the future.

The majority of open source developers spend most of their time on closed code development. Most open source developers work primarily on internal or closed code development within companies, so they are quite familiar with closed code.

Open source products are often sold as part of a bundled sale, which includes closed code products.

Large organizations often purchase the top level of the software and service stack from a major closed code vendor.

Their primary purchase might be outsourcing or other services from IBM Global Services, Accenture, or CSC; software from IBM, BEA, Oracle, or SAP; hardware from IBM, HP, or Dell, with Linux and other open source products included in the overall sale.

It is common for companies to take open source projects, add a layer of additional functionality in closed code, offer support for both their enhancements and the open source base, and charge a fee. This describes IBM WebSphere, Red Hat, and some other distribution companies.

Dual Licensing

Some products are dual licensed. They are available with either an open source license or a commercial license. Examples of such products are:

- Qt, from TrollTech, the GUI toolkit used by KDE
- MySQL, from MySQL AB, the database server
- Berkeley DB, from SleepyCat Software, the embedded database Program

The dual license allows these companies to offer open source products to those who are developing open source software, or to individual end users. Depending on their intentions or organization, others may be required to pay for a commercial license.

Other Intellectual Property Issues

Intellectual property (IP) refers to creations of the mind, such as inventions; literary and artistic works; designs; and symbols, names and images used in commerce.

It is important not to confuse these, so it is always helpful when looking at a particular claim to see which of these it is based on rather than using loose terminology such as “intellectual property.”

Open source software should not be confused with public domain. Open source uses copyright and licensing law to enforce intellectual property rights, essentially like any other licensing agreement but with a particular purpose in mind. Including public domain as an option, then, our distribution choices come down to these:

Closed code we grant a license with restrictions that we specify, typically including relicensing and publication of proprietary information, and retain ownership.