CHAPTER 1. INNOVATION IN THE SOFTWARE SECTOR: ECONOMIC PROCESSES

OECD PROJECT ON INNOVATION IN THE SOFTWARE SECTOR

This document is submitted for information and discussion under Item 6 a) of the CIIE agenda. It does not incorporate feedback received from the meeting of the Software Advisory Expert Group that took place in Tokyo on 7 October 2008.

Contact: Douglas Lippoldt, E-mail: douglas.lippoldt@oecd.org Tel. (33-1) 45 24 19 26
Piotr Stryszowski E-mail: Piotr.Stryszowski@oecd.org Tel. (33-1) 45 24 91 30 and
Jeoung-Yeol Yu E-mail: Jeoung-Yeol.Yu@oecd.org Tel. (33-1) 45 24 98 74
CHAPTER 1
INNOVATION IN THE SOFTWARE SECTOR: ECONOMIC PROCESSES

Introduction and main findings

1. Software has become a virtually ubiquitous phenomenon in modern economies, playing a vital role in a growing number of fields with respect to products and processes. There is a huge variety of possible functions that software can offer – from core codes and operating systems to applications that provide various functionalities to end users. Embedded software has become a key component of various products and systems. Business applications are important input in a growing number of industries.

2. Innovation in the software sector has twofold importance for OECD economies: software is not only a highly innovative and economically important sector in its own right, but software is often also an important element of innovation in other sectors. Many process and structural innovations depend heavily on organisational changes that are facilitated by software innovations. Such changes can be a major driver of productivity improvements. Thus, the expected economic impact of software innovation is likely to be much greater than what is observed by solely examining capital investment in the sector.

3. Moreover, software innovation has gone global. Technological developments, in particular in information and communication technologies, and the roll-out of high-speed broadband communications networks, have hugely increased the scope for globalisation of software activities. As a result, production and consumption of software can take place in different locations, internationalising software research and development activities and creating new market segments such as “software as a service”.

4. Drawing on current literature, OECD and other data, input from an Advisory Expert Group, the results of an OECD business questionnaire (Box 1) and other resources, the Secretariat has conducted this analysis over a number of months. The following assessment provides some important policy-relevant insights into the nature of innovation in the software sector, pointing to the following main findings:

- *Software is poorly captured by existing statistics* – Since software is a non-physical good and is evolving rapidly to take on new forms (e.g. software as a service), it is often poorly captured in economic statistics.

- *Available evidence points to a high degree of dynamism in the sector* – The software sector is dynamic in the pace and extent of its growth and innovation. The dynamism is reflected in the development of traditional software firms and the emergence of many innovative start-ups. It is associated with the emergence of new approaches to software business and interaction with partners in other industries, among other factors.

- *Numerous modes of software innovation are employed, some with software-specific features* – Innovation in the sector proceeds in an incremental and cumulative fashion, capitalising on the characteristics of software as an intangible, digital product. In some cases, traditional linear modes of innovation are employed, in others collaborative approaches are employed (some of which are software-specific).
• **Business models in the sector interact with the innovation processes** – Some models are built around the innovative process (e.g. open source approaches), while others have been made possible by recent innovation in information and communication technologies (e.g. cloud computing). In some cases, the vendor-customer relationship resembles a strategic partnership, whereby development across the stakeholders is mutually reinforcing, yielding further innovation. The range of business models changes in an on-going fashion.

• **Software innovation is human capital intense** – The availability of trained and creative workers is a key element in the software innovation process; human capital is a crucial input. Depending on the nature of the innovation, the physical capital requirements can be relatively modest.

• **Environmental factors are important to the software innovation process** – The nature of the intellectual property regime, technical standards, legal and regulatory requirements and other environmental factors influence the ability of software firms to optimise their processes in favour of innovation. (An OECD business questionnaire provides some insights in this regard. See Box 1.)

5. Taken together these findings provide some useful pointers for consideration in development of subsequent policy frameworks. They do not constitute recommendations. They do, however, point to both the need for caution in government policy so as to avoid interfering with the fruitful innovation processes, while at the same time delivering appropriate policy support in key areas of public interest (e.g. human capital development).

---

**Box 1 – OECD Business Questionnaire**

The OECD project “Innovation in the Software Sector” was carried out in co-operation with an Advisory Expert Group (AEG) comprised of representatives from business, academia and government, with particular expertise in range of relevant areas. With the assistance of the AEG, during the second quarter of 2008, the Secretariat circulated a detailed (20 page) questionnaire to businesses engaged in software innovation. Responses were received from 27 firms located in Germany, Japan, Lebanon, Mexico, Poland, Spain, Switzerland, Turkey and the United States, ranging in size from USD 1 million in turnover to multiple billions and from a few dozen employees to 10,000s. Of these firms, about ¾ were independent enterprises, with more than 2/3 having subsidiaries. About 2/3 had developing and selling software as their main activity. The business questionnaire was not intended to be statistically representative, but rather to provide illustrative insights from a diverse group of enterprises. Results are discussed throughout the Chapter.
The Nature of Software

What is software?

6. Generally, the term software is used to describe the digital instructions and operating information that are contained in programs serving to guide machines – especially computers – in implementing desired operations (e.g. process data or interact with peripherals).\(^1\) Software is created from source code, which consists of sequences of statements and declarations written in special computer programming languages. The source code is usually held in one or more text files, a large set of which may be organised into a directory tree (also called a source tree). Source code is usually converted into a machine-executable digital format using special compiler programs.\(^2\) End users generally get only the executable files; the source code does not necessarily need to be revealed to the user. Open source software (OSS) either includes or permits ready access to the human-readable source code in order to facilitate further modification or reuse, subject to certain conditions (e.g. licensing).\(^3\)

7. Two key features of software are that it is disembodied and complementary to hardware:\(^4\)

- **Disembodied character** – Software is a non-physical good. It is a digital sequence of commands stored on physical carriers, such as a hard drive, memory disk or compact disk. Software can be recorded and stored in either a temporary or permanent fashion. The main implication of the non-physical form of software is that once developed, a software product can be replicated with relatively little cost and effort. Although a software supplier may incur some costs in delivery of products to the market such as direct cost of delivery (e.g. packaging), customer support or provision of software updates, the marginal cost of production for a given version of a software product is generally modest and may be close to zero in some cases. The disembodied character of software also facilitates digital transmittal (e.g. via the Internet), which can facilitate and expedite the delivery from the producer to users.

- **Complementarity with hardware** – Hardware provides the physical support for software and without it software is merely a set of instructions that cannot be executed by a machine. Similarly, modern computer hardware requires software to function and is typically incapable of providing any advanced functionality in the absence of software applications. As a result, there are strong linkages and complementarities between hardware and software markets. A given software product may be hardware agnostic, able to run on various types of hardware. Some software products, on the other hand, are especially designed for specific hardware products. The

---

\(^1\) The software field has a particular jargon and definitions for particular terms may vary, which can be confusing. A variety of easy-to-understand glossaries exist to help guide those new to the field. See, for example, Bixler and Bergman (1997) or the UNESCO Learning Without Frontiers web site: [http://www.unesco.org/education/eduprog/lwf/doc/portfolio/definitions.htm](http://www.unesco.org/education/eduprog/lwf/doc/portfolio/definitions.htm).

\(^2\) In some cases, alternatively, the source code can be executed directly using an intermediate computer program called an interpreter.

\(^3\) The term open source software (OSS) is often used loosely to refer to the three dimensions of openness. First, it points at the openness of the source code of software, as opposed to the case of closed source where the source code is not revealed publicly. Second, openness can refer to the openness of gratis distribution. Third, openness can refer to openness to improvement. In the context of this study the term OSS also encompasses so-called “free software” (a definition of the latter concept can be found on the GNU project web site: [http://www.gnu.org/philosophy/free-sw.html](http://www.gnu.org/philosophy/free-sw.html)).

\(^4\) Not all electronic content can be classified as software. In particular, it is important to distinguish software from data. While a software program commands a physical device (hardware), data contain only given information and are not designed to control or instruct any hardware.
rapid pace of technological change in hardware\(^5\) can influence software development; for example, programmers may move quickly to exploit expanding capabilities in new generations of hardware.

8. Such characteristics of software influence the profile of innovation in the sector including, for example, certain aspects of its nature and pace:

- **Cumulative development** – Most software development and change-in-added-value occurs through a multi-stage process that builds on previous innovative steps (FTC, 2003). The accessible, disembodied nature of software facilitates transfer, reuse and modification (e.g. revision, addition, linkages) of code and is conducive to such a process. Thus, cumulative development plays a particularly notable role in the case of software products, even though it is not wholly unique to the sector.

- **Short lifecycles, especially for specific product versions** – The disembodied character of software and reliance on cumulative, incremental development processes (Harter et al., 2000) together contribute to the relatively-frequent release of new products and product enhancements. As FTC (2003) points out, most software products experience much shorter lifecycles than other traditionally manufactured goods.

**Software structure and function**

9. Software is developed by many different entities including firms that specialise primarily in software, firms outside the sector that produce software in addition to their main lines of business, public sector and academic institutions, non-governmental organisations and individual programmers. The diversity of the stakeholder perspectives and the expanding nature of software itself, with deployment across most facets of life, make it difficult to propose one, unified and unambiguous taxonomy to classify its key dimensions.\(^6\)

10. One approach is to sketch the various key types of software in terms of their function and relationship in the architecture and geography of systems. While there is bound to be some overlap in the various categories under such an approach (e.g. a particular function may be performed alternatively by more than one type of software, each in a variety of configurations), this mapping of software permits a general description and basic analysis.

**Software Architecture**

11. Architectural approaches sometimes view software as a stack ranging from the machine interface to the user interface. While the architecture is changing and a multidimensional view is now required, software structure remains layered. One important dimension of change in recent decades can be found in the geography of software. While software still operates locally, it is increasingly structured to operate in a distributed or remote fashion.

\(^5\) For example, Moore’s Law holds that the density of transistors on an integrated circuit doubles every 18 months, growing as rapid geometric progression and providing more processing speed and possibly expanded functionality. Other dimensions of hardware are also experiencing rapid change and may reach a similar pace of evolution as a consequence of increased integration (IEEE, 2006).

\(^6\) For example, one traditional classification of the sector used by some public agencies for statistical purposes, focussed on just three types of software that could be readily measured (packaged, in-house and custom); but in the current environment, this is inadequate for describing the variety of new forms and modalities operating in the sector such as software as a service (described below).
12. A basic schema for software architecture considers three layers: applications, operating systems and middleware (Figure 1). Under this approach, software applications can be seen as programs that offer functionality directly sought by users, they may reside locally or remotely. Operating systems reside locally and manage hardware resources, serving as the software interface relating certain user requests to the hardware and providing the structure enabling other types of software to run. The third layer is middleware, which is employed in cases where processes run remotely. Middleware bridges among applications and operating systems by providing a standardised interface that can permit interoperation of applications on different platforms or written in different languages. Applications generally interact with other software via application program interfaces (APIs) that provide a means for them to communicate via specific language and message formats. These three elements of software architecture (applications, operating systems and middleware) are discussed in the following sub-sections.

Figure 1: One View of Distributed Software Architecture

![Diagram of software architecture][1]


Applications

13. Software applications are programs, usually designed to deliver sets of functionality to end-users and, sometimes, also to other applications. They are essentially located on top of the operating system and generally cannot function without the presence of an operating system to manage their interaction with the components of the hardware device where they reside. Common types of applications for household and office users include software for text processing, electronic spreadsheets, database management, instant messaging, email, web browsing and desktop publishing, among many others. Specialised applications can also be specific to relatively narrow task sets (e.g. machine control in manufacturing) or to particular industry needs (e.g. film production software). Software applications are often bundled by developers in order to provide a consistent user experience across ranges of functionality; these are referred to as application suites. Within these suites, applications can interact with each other in order to facilitate data transfer and other user conveniences. Similarly, developers may use APIs to join simple applications together into composite applications or mash-ups that provide a greater range of functions in an integrated fashion. While applications may reside locally on the user’s computer system, increasingly they are accessed remotely, including via the Internet; in some cases, they are offered as a service that is paid for based on usage or through a subscription fee.

---

7 PC Pro magazine offers research on an list of applications software, which is illustrative of the diversity of application software: barcode label, communications, consumer applications, contact management, diagnostic and testing, enterprise applications, geographical information, graphics, health care, industrial applications, insurance applications, interactive media applications, internet applications, legal applications, mathematical, network applications, office applications, portal applications, project management, publishing, scientific applications, simulation and analysis and wireless applications. See: [http://research.pcpro.co.uk/rlist/term/Applications-Software.html](http://research.pcpro.co.uk/rlist/term/Applications-Software.html) (last accessed on 26 July 2008).
Operating Systems

14. In serving as the interface between users, other software and the hardware, operating systems perform a number of core functions (Silberschatz, Galvin and Gagne, 2005). Operating systems:

- execute user programs,
- employ computer hardware in an efficient manner,
- bring convenience to user interaction with the computer.

They are the backbone of the processes that users demand, residing in many computing environments such as supercomputers, mainframes, servers, desktops, workstations, handheld devices, real-time, and embedded systems. Examples from desktop computing may include such operating systems as Windows, OS X or Linux, among many others.

15. According to Silberschatz et al. the specific functions of operating systems can be grouped into two large sets functions: i) those related to process management and ii) those related to memory management. Process functions involve actions such as process execution, management, supervision, interruption or destruction. Memory management involves supervision of hardware memory capacities. The latter function is important because programs function on a time-sharing basis and memory must be allocated in a manner such that programs do not unduly interfere with each other. Box 2 itemises the key activities of the operating system in more detail.

<table>
<thead>
<tr>
<th>Box 2 – Operating System Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Management of the processor</strong> – allocation of the processor resource among the different programs using a scheduling algorithm, according to the desired objective.</td>
</tr>
<tr>
<td><strong>Management of the random access memory</strong> – allocation of the memory space to each application and, where relevant, to each user. If there is insufficient physical memory (e.g. random access memory), the operating system can create a memory zone on the hard drive, known as “virtual memory”. The virtual memory lets users run applications requiring more memory than is available on the system. There also can be cache memory where frequently executed instructions and data reside.</td>
</tr>
<tr>
<td><strong>Management of input/output processes</strong> – integration and control of program access to material resources via drivers (also known as peripheral administrators or input/output administrators).</td>
</tr>
<tr>
<td><strong>Management of execution of applications</strong> – allocation of resources required for applications to operate. An application that is not responding correctly can be “killed”.</td>
</tr>
<tr>
<td><strong>Management of authorisations</strong> – handling of security relating to execution of programmes by guaranteeing that the resources are used only by programmes and users with the relevant authorisations.</td>
</tr>
<tr>
<td><strong>File management</strong> – reading and writing in the file system and the user and application file access authorisations.</td>
</tr>
<tr>
<td><strong>Information management</strong> – provision of indicators that can be used to diagnose the operation of the computer.</td>
</tr>
</tbody>
</table>


Middleware

16. Middleware, often referred to as business integration software, can be seen as a software intermediary between an operating system and applications which may reside on multiple machines. Middleware is used to facilitate application development by providing common programming abstractions, by masking the heterogeneity and the distribution of the underlying hardware and operating systems, and
by hiding low-level programming details. Middleware development, production, distribution and
innovation occur across a range of environments (e.g. with respect to the licensing approaches). It permits
applications to (Schreiber, 1995):

- locate transparently across the network, providing interaction with another application or service,
- be independent from network services,
- be reliable and available, and
- scale up in capacity without losing function.

Generally, the delivery of integrated services, which may be implemented across a diverse range of
computing environments, is done in a fashion that is seamless in terms of end-user experience.

17. Concretely, middleware can enable systems to deliver services such as:

- **Distributed system services** – including communications, program-to-program and data
  management services across a number of locations;
- **Application enabling services** – giving applications access to distributed services and the
  underlying network;
- **Management services** – enabling applications and system functions to be continuously monitored
  to ensure optimum performance of the distributed environment.

Across the various systems and functions, middleware must operate securely, ensuring services such
as communication encryption, user authentication and access control, as necessary.9

Beyond the Basics

18. The basic categories listed above do not provide a comprehensive picture of the full scope of
software and also mask many sub-categories of software within each of the headings. Beyond these basic
categories, a variety of additional ones may be specified from a variety of perspectives. The following
section describes several categories that are of particular interest from the perspective of software
innovation within the scope of the OECD project.

Cloud Computing

19. Cloud computing is a nebulous, software-based concept that goes beyond consideration of the
functionality of the programs to include the manner of delivery, location and user experience, among other
dimensions. Generally, the concept refers to application, platform or utility services accessed by users over

---

8 Examples of middleware initiatives include Open Software Foundation's Distributed Computing
Environment (DCE), Object Management Group's Common Object Request Broker Architecture
(CORBA), and Microsoft's COM/DCOM (Component Object Model (COM), DCOM, and Related
Capabilities), http://www.sei.cmu.edu/str/descriptions/middleware.html (last accessed 19.05.2008).

9 From information on the 1st International Conference on Middleware Security (2008), available here:
the internet in real time from the global ecosystem of service providers. Cloud computing can be massively scalable, readily accessible, highly reliable, and cost effective, with low barriers to entry to new providers of services (Johnston, 2008). Often, the cost of using cloud-based services is comparatively low because cloud computing enables the providers to respond flexibly to market demand from a global pool of users for a particular software service, thereby exploiting economies of scale. Indeed, some advocates foresee availability of cloud computing power at utility prices, similar to electricity or water.

20. Although it is in an early phase of development, cloud computing is already permitting users to share resources in a generally fast and efficient way, accessing the extraordinary computing power available from large providers (Carey, 2008). It is a way to increase capacity or add capabilities without investing in substantial new internal computing infrastructure, software licensing and training. It can help firms reduce the need to acquire, configure and administer hardware and software, which can often be the source of project delays or failures. Generally, it can enable users to avoid the cost of down time or underutilization that may plague some in-house systems. Often, the services are paid on the basis of usage (e.g. pay-per-use) or subscription, though some may be indirectly-paid via advertising or sales of third-party products.

21. For some software innovators, references to the cloud go beyond the delivery of software services to the market. The cloud is also a resource, whereby innovators can pose difficult questions to a large resource base (e.g. via blog) and draw on the collective wisdom among experts in particular internet communities to resolve them (OECD, 2008b).

22. Cloud computing is just now taking off for consumers, some of whom, for example, are turning to such sites as Google Apps (office suite) or Apple’s MobileMe (storage and synchronization services) for access to specific computing services. However, some enterprises are more advanced in the process of virtualisation as they strive to develop agile, scalable infrastructure. Others are turning to on-line platforms as a vehicle to develop and market applications (e.g. via Salesforce.com), which may enable them to meet their own computing needs while also potentially monetizing applications they develop themselves through sales to other enterprises. Major players in cloud computing for enterprises include Amazon Web Services, IBM and Google, among others.

23. Though the concept of cloud computing is subject to further development, it generally encompasses such services as (Knorr and Gruman, 2008):

- **Software-as-a-service (SaaS)** providers deliver access to applications and other services via the Internet browser (discussed in more detail below).

- **Platform as a service** providers offer software development environments, as well as marketing and delivery channels.

- **Web services in the cloud** deliver application program interfaces (APIs, discussed below) that enable developers to focus on building the software functionalities they wish to deliver, while capitalizing on off-the-shelf functionalities available from the service provider (e.g. such as transaction management).

- **Utility computing** constitutes a virtual resource pool available over the web for supplemental data storage, computing capacity, input and output management, among other functions.

10 Sometimes, firms or on-line communities make reference to closed, cloud-based systems. But, for purposes of this paper, the cloud is seen as a universal concept encompassing the entire Internet. Bounded cloud-based systems are seen as sub-sets of the whole.
- **Managed service provision** (MSP) refers to an application operating via the Internet and interacting with the enterprise user’s locally functioning information technology processes rather than offering functionality to consumers. Examples of MSP include virus scanning service for e-mail or an application monitoring service.

- **Service commerce platforms** combine SaaS and MSP to deliver a service hub giving specific users access to controlled environments (*e.g.* with expense management in mind), that enable them to then purchase services such as travel or secretarial services.

- **Internet integration** refers to integration of cloud-based services such as might be offered to SaaS providers on a business-to-business basis or integrated solutions to consumers drawing on various SaaS providers.

Software-as-a-Service

24. Under the concept of SaaS, the focus shifts from on-premises delivery to the customer of a software package towards delivery to the customer of a software service. As with the broader category of cloud computing, there is no standard definition for the concept of SaaS. Gartner (2007) describes SaaS succinctly as “software that's owned, delivered and managed remotely by one or more providers”. A more expansive definition states that “Software as a Service is time and location independent online access to a remotely managed server application, that permits concurrent utilization of the same application installation by a large number of independent users (customers), offers an attractive payment logic compared to the customer value received, and makes a continuous flow of new and innovative software possible” (Lassila, 2007). Although SaaS is still in a relatively early stage of adoption in the market, the influx of investment in infrastructure and entries into the market provide an indication that the SaaS model is gaining traction. Table 1 presents an illustrative list of key differences between SaaS and more traditional software-as-an-application approaches.

### Table 1. Traditional Software-as-an-Application versus Software-as-a-Service

<table>
<thead>
<tr>
<th>Software-as-an-Application</th>
<th>Software-as-a-Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users pay upfront for the license</td>
<td>Users &quot;pay as you go&quot;</td>
</tr>
<tr>
<td>Dedicated instance of software is installed on user's hardware</td>
<td>Software is managed and maintained by SaaS provider</td>
</tr>
<tr>
<td>Users are responsible for deployment, operation and maintenance of the IT infrastructure required for the application</td>
<td>The SaaS provider is responsible for the infrastructure</td>
</tr>
<tr>
<td>Users are responsible for upgrading software</td>
<td>The SaaS providers upgrade software automatically</td>
</tr>
</tbody>
</table>


---

11 A similar definition was proposed by McNee (2007) who defines SaaS in terms of the functionality of the application or utility infrastructure software that is provided by a SaaS supplier over the network.

12 A related concept is service oriented architecture (SOA), whereby enterprises can structure their computer systems in modules that can be accessed in various combinations and by various users as necessary to deliver services. IBM, for example, offers services to transform business legacy systems using SOA approaches as a way to update them and extract value.

13 For example, a March 2008 abstract from Forrester points to rapid growth in this area: [http://www.forrester.com/Research/Document/Excerpt/0,7211,44254,00.html](http://www.forrester.com/Research/Document/Excerpt/0,7211,44254,00.html).
25. **SaaS providers** tend to have the following characteristics (IDC as cited in SIIA, 2001):

- **Application centric** – SaaS providers generally offer access to, and management of, an application that is commercially available.

- **Sale of access to applications** – Customers gain access to a new application environment without making up-front investments in the application license, servers, people and other resources. The provider either owns the software or has a contractual agreement with the software vendor to license access to the software as part of the offering. Generally under such arrangements, the SaaS provider has responsibility for the software upgrades and maintenance.

- **Centrally Managed** – The application service is managed from a central location rather than at customers’ sites.

- **One-to-many service** – SaaS tends to be designed to be a one-to-many offering.

- **Delivers on the contract** – There are many partners working together to provide a SaaS solution. The provider is the firm that is responsible, in the customer's eyes, for delivering on the customer contract for provision of the application service.

A particular form of the SaaS model is the ASP model (Application Service Provider), a term said to have been coined by the market intelligence firm IDC.\(^{14}\) In the ASP model, the customer pays the software vendor for the application software license, while the service provider hosts and manages that application on its servers; the application is then accessed by the user through direct connections or via the Internet.

There are numerous functions offered by SaaS providers including, for example, customer relationship management (CRM), human capital management (HCM) and enterprise resource planning (ERP):

- **CRM functions** include the design and control of the pre-sale and post-sale activities of a company in relation to their customers. They include various aspects of dealing with customers such as marketing, call centre interaction, sales and technical support.\(^{15}\) Providers aim to deliver improved integration within and across these areas.

- **HCM functions** aim to align business and individual performance goals and to consolidate multiple human resource transactional systems for efficiency and global workforce visibility (Forrester, 2006).

- **ERP functions** include software to manage the acquisition and deployment of resources across an enterprise.

26. In some cases, providers combine their SaaS role with other offerings such as integration services (e.g. business and supply-chain integration). Gartner (2006) found that “…the majority of integration service providers will offer some form of business activity monitoring (BAM) capability (for example, giving users visibility into the execution of the order-to-pay process, linking payments to their

---

\(^{14}\) According to the SIIA (2001), the term ASP was coined by IDC and referenced in the paper "ASPs Are for Real … But What's Right for You?" an IDC White Paper (referenced at this web site, for example, but no longer available: [http://whitepapers.zdnet.co.uk/0,1000000651,260001630p-39000471q.00.htm](http://whitepapers.zdnet.co.uk/0,1000000651,260001630p-39000471q.00.htm)).

\(^{15}\) PC Magazine, (2008), [http://www.pcmag.com/encyclopedia_term/0,2542,t=CRM&i=40485,00.asp](http://www.pcmag.com/encyclopedia_term/0,2542,t=CRM&i=40485,00.asp), (last accessed 04.07.2008).
associated invoices and to their associated purchase orders, and allowing users to generate alerts on payments). In addition to expanded supply chain integration capabilities, many providers of hosted integration services also take the role of application service providers, with some form of hosted applications.”

Embedded software

27. Embedded software generally resides on a long-term basis in hardware units other than computers, where it is used to control various product components. In contrast to software operating, for example, on a standard desktop computer, such software is almost never directly manipulated by a user, though user input may be required to specify actions or select options among the various functionalities. That is, such software is usually self-contained and not subject to user modification. Consequently, it must be “extremely reliable, very efficient and compact, and precise in its handling of the rapid and unpredictable timing of inputs and outputs”. Embedded software is on the way to becoming ubiquitous in modern economies, being found in a very broad range of electronic products and systems. Examples of industries with particularly heavy use of embedded applications include the automobile industry, mobile phones, robotics, telecommunication systems, medical devices and consumer electronics.

28. Recent studies indicate that the embedded software market is rapidly growing and coming to play an important role outside of the traditional software sector. Van Genuchten (2007) points to its importance for the electronics sector and foreshadows its future expansion in other sectors. A more quantitative insight is presented by METI (2007), through an enterprise survey on embedded software. The results indicate that the market for embedded software development in Japan grew during 2006-2007, for example, at an annual yearly rate of 19.8% attaining a level of JPY 3,72 trillion (ca. USD 30 billion), rapidly expanding in usage across various sectors of the economy.

29. Embedded software differs from other types of software in that it often does not appear on product invoices (van Genuchten, 2007). The reason is that this type of software is generally sold as an integral part of the hardware products where it resides. However, there are some indications that embedded software developers are seeking to improve their business models by finding new ways to market, reuse or repurpose their embedded software and thereby generate direct revenues from it (discussed in more detail in Chapter 3).

Software: Scale in the National Economy

30. Generally, the market demand for software appears to be growing quite rapidly. This is perhaps not surprising, given the rise in the number and complexity of individual systems that employ software in an expanding range of applications. The growth is fuelled by such factors as globalisation, technological progress (especially in information and communication technologies, ICT) and deregulation (e.g. expanding the availability of radio spectrum for commercial use) and is sustained through the application of innovations and entrepreneurship to solve specific problems. Despite concrete indications of growth in the software sector, precise assessment of software production and distribution is especially challenging in comparison to markets for more traditional products (Box 3).

31. Definitions of the term “software sector” may vary in the scope of the activities that they cover. In this paper, the term “software sector” is defined loosely to include the traditional “software industry”

---

16 This quote is from the description of embedded software provided by the Embedded Software Center of the University of Texas at Dallas, http://esc.utdallas.edu/overview.html (last accessed 01.08.2008).

17 The METI survey covered enterprises in the field of embedded software and included valid responses from 311 operational departments in 293 enterprises, 525 projects and 769 embedded software engineers.
(i.e. companies or institutions that primarily deal with development of software such as Microsoft, Adobe or Mozilla), as well as the parts of other industries that are involved in software development. Thus, the perspective employed here includes elements of such industries as finance, electronics and government, among others, as well as some academic or non-profit institutions.

32. OECD statistical measures, where they have covered specific types of software, have tended to track three main categories, whereby the first two represent software that is purchased from external suppliers (Ahmad, 2003; Lequiller et al., 2003):

- **packaged software** – software that can be directly acquired “off-the-shelf” from a retail store or via the Internet,
- **custom software** – software prepared specifically for a customer by an external supplier, and
- **in-house software** – software that is developed within a company or organization to suit its own needs (e.g. for use embedded in an electronic product produced by the firm).

Statistics based on these categories often omit certain segments of the software sector (e.g. in relation to some open source software) due to an approach that only covers software that has a price or software whose cost of creation or acquisition can be established based on accountancy techniques. In some specific cases, the omitted coverage may be significant. For this reason, the following section – which draws on available data on investment and revenue in order to substantiate the scale of the sector in the economies of OECD countries – likely understates its true scale.

**Software Investment**

33. Ahmad (2003) presents a cross-country comparison of software sector investment with respect to purchased and in-house software (Table 2). The results indicate that, at the time of his analysis, investment in these types of software was already equivalent to sizable shares of Gross Domestic Product (GDP) in the selected OECD countries, equating to between 0.5% and 2.7% of GDP. In terms of GDP shares, Sweden had a sizable lead over the other countries shown in the table, with the United States placing second. With respect to the specific types of software, Sweden led in terms of investment in purchased software and tied with Canada in terms of investment in in-house software.

34. The OECD Productivity Database (OECD, 2007) presents estimates of the value of software investments for OECD member countries drawing on enterprise data collected by national statistical offices. The evolution of investment in software relative to non-residential gross fixed capital formation from 1980 is shown in Figure 2. Overall, investment in software demonstrated substantial increases for most OECD member countries for which data were available, with some variation by year and country. These figures are all the more impressive, given the likely underestimation of the scale of investment in software (Box 3).

---

18 According to OECD, gross fixed capital formation as defined by the European System of Accounts “consists of resident producers’ acquisitions, less disposals, of fixed assets during a given period plus certain additions to the value of non-produced assets realised by the productive activity of producer or institutional units.” OECD on-line *Glossary of Statistical Terms*, available here: [http://stats.oecd.org/glossary/detail.asp?ID=1173](http://stats.oecd.org/glossary/detail.asp?ID=1173) (accessed on 02.08.2008).
Table 2. Software investment as percentage of GDP

<table>
<thead>
<tr>
<th>Country</th>
<th>Purchased Software</th>
<th>In-House Software</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Australia</td>
<td>1.1</td>
<td>0.7</td>
<td>1.8</td>
</tr>
<tr>
<td>Canada</td>
<td>0.8</td>
<td>1.0</td>
<td>1.8</td>
</tr>
<tr>
<td>Czech Republic</td>
<td>1.0</td>
<td>n.a</td>
<td>n.a</td>
</tr>
<tr>
<td>Denmark</td>
<td>0.8</td>
<td>0.4</td>
<td>1.3</td>
</tr>
<tr>
<td>Finland</td>
<td>n.a</td>
<td>0.4</td>
<td>n.a</td>
</tr>
<tr>
<td>France</td>
<td>0.6</td>
<td>0.8</td>
<td>1.5</td>
</tr>
<tr>
<td>Greece</td>
<td>0.3</td>
<td>0.2</td>
<td>0.5</td>
</tr>
<tr>
<td>Italy</td>
<td>0.5</td>
<td>0.4</td>
<td>0.9</td>
</tr>
<tr>
<td>Japan</td>
<td>1.3</td>
<td>0.6</td>
<td>1.9</td>
</tr>
<tr>
<td>Netherlands</td>
<td>0.9</td>
<td>0.9</td>
<td>1.8</td>
</tr>
<tr>
<td>Spain</td>
<td>0.6</td>
<td>0.3</td>
<td>0.9</td>
</tr>
<tr>
<td>Sweden</td>
<td>1.7</td>
<td>1.0</td>
<td>2.7</td>
</tr>
<tr>
<td>United Kingdom</td>
<td>0.8</td>
<td>0.8</td>
<td>1.7</td>
</tr>
<tr>
<td>United States</td>
<td>1.2</td>
<td>0.9</td>
<td>2.1</td>
</tr>
</tbody>
</table>

Notes: n.a. = not available. Based on national data adjusted to improve comparability.

Source: Ahmad (2003).

Figure 2. Share of software investment in total non-residential gross fixed capital formation, %

Box 3 – The Challenge of Software Measurement

Clear and reliable statistics are an important foundation for economic assessment of any industry. Yet, in the case of the software sector, measurement (e.g. determination of actual economic value) and analysis of the data are not easy tasks. There are several main reasons for this:

Software is an intangible good – A given software product often reflects the culmination of a very large and costly research and development effort (Chapter 2). Yet, as a digital product, software features an extremely low cost of duplication. Thus, marginal costs of production may serve poorly as a starting point for an exercise aimed at determining the economic value of an additional unit of software.

Non-traditional approaches to development, production and distribution – A significant portion of software development, production and distribution may not be separately identified in enterprise costs and revenues (being bundled with other activity), or may occur outside of the standard commercial environment. There may be no monetary payment (e.g. wages or price) tracked separately for use in directly valuing the software. That is, the software costs or revenues may be embedded in other aspects of the operation (e.g. revenue may come indirectly such as via advertisements delivered via the software or from services provided for support and maintenance of software). In other cases, development may advance partly through non-monetary incentives or external collaboration. Thus, for some software products there are no readily available proxies for use in economic valuation.

Variability in treatment of software statistics – Even for the types of software (e.g. packaged software) or dimensions of software (e.g., investment in software acquisition) that are most readily tracked, there is some variation among businesses and nations in the treatment of the statistical measures. For example, accounting standards may differ in their disclosure requirements for businesses or the handling of capitalisation of software investment in national accounts may vary.

International comparability of price levels – Controlling for comparative price levels across countries is a very difficult task, particularly given the intangible nature of software, a rapidly changing market environment, and the lack of observable market price for some types of software. As Ahmad (2003) points out, inconsistencies in price indices may be a significant factor in international comparisons.

Data on software investment may be subject to particular biases, due to the effects of the accounting treatment of software investment and valuation approaches in investment surveys (Ahmad, 2003).19

Accounting treatment of software investment may introduce downward biases – In the United States, for example, most of the software products are capitalized directly, whereas in the United Kingdom most of the software is capitalized indirectly (i.e. when purchased as a bundle with PCs), and hence the investment is recorded as investment in hardware, not software. In another example, French statistics report a high proportion of software expenditure as “software consultancy”, whereas other countries may classify such expenditure as “capital investments”. Ahmad (2003) also points out that tax schemes can create incentives for firms to understate the value of in-house software.

Investment surveys vary in their approach to valuation of in-house software – Generally, these estimates value such software based on the estimated costs of software development (e.g. wage costs). However, there is variation across countries in the estimation techniques. For example, there is no consistent statistical treatment of wages and salaries across countries (e.g. which may – or may not – include employers’ social contributions). Moreover, some countries do not capitalize in-house software if it is used for further development of other software products. Moreover, evidence from one OECD study suggests that many firms do not capitalise in-house (own-account) production of software at all (OECD, 2001).

---

19 This is despite progress in the System of National Accounts (1993 manual), which moved to better reflect software, stating: “Computer software that an enterprise expects to use in production for more than one year is treated as an intangible fixed asset. Such software may be purchased on the market or produced for own use. Acquisitions of such software are therefore treated as gross fixed capital formation. Software purchased on the market is valued at purchasers’ prices, while software developed in-house is valued at its estimated basic price or at its costs of production if it is not possible to estimate the basic price. Gross fixed capital formation in software also includes the purchase or development of large databases that the enterprise expects to use in production over a period of time of more than one year. These databases are valued in the same way as software” (paragraphs 10.92 and 10.93).
Software: Contribution to Dynamics of the Economy

35. The growing importance of software in OECD economies is underscored by its increasing contribution to capital services. Capital services represent the flow of productive services provided by an asset that is employed in production.\footnote{According to the OECD’s on-line \textit{Glossary of Statistical Terms}, capital services “reflect a (physical) quantity, not to be confused with the value, or price concept of capital. Capital services are the appropriate measure of capital input in production analysis.” The glossary is available here: \url{http://stats.oecd.org/glossary/detail.asp?ID=270} (accessed on 02.08.2008).} Based on data from the OECD Productivity Database (OECD, 2007), the contribution of the software industry to the growth of capital services can be calculated for the period 1980 to 2004. During this time, software’s contribution rose continuously in all OECD countries for which data were available, particularly the United States and Sweden (Figure 3).

\textbf{Figure 3.} Software’s contribution to growth of capital services, 1980-2004

(average annual change, %)

![Software's contribution to growth of capital services, 1980-2004](image)


36. Disaggregating the capital services data by 5 year time periods, reveals that the generally positive growth in software’s contribution was positive in nearly all of the countries and time periods for which data were available (Figure 4). For a number of the countries, the second half of the 1990s was a time of particular strength in this indicator, with Sweden, the United States and France each experiencing increases of greater than 1% per year. The chart also reveals significant period-to-period variability in the indicator in some countries.
37. The dynamism of the software sector is also reflected in its contribution to added value (i.e. the contribution of software as a factor of production in augmenting the value of a final product). According to one OECD study (2006), the software sector reported the highest growth rates of value-added among all the ICT industries. The nominal value added in the broadly-defined software sector represents around 1.5% to 3% of business sector value added in OECD countries. While the share remains relatively small, it has been increasing regularly.

38. The highly dynamic character of software is also confirmed by recent data on financial indicators of the top 500 software companies (Software Magazine, 2008). For example, these data point to the high growth rates of overall revenues during the period 2006 to 2007 including from activity directly related to software (Table 3). Looking at just the top 50 companies, average revenue increases are somewhat smaller, which implies that the smaller firms in the group contribute disproportionately to the growth. This growth comes on top of an already substantial revenue base for the leading firms (Box 4).

39. Further evidence of the dynamism and innovative character of the software sector comes from White et al. (2004) who consider developments in software prices. The authors analyzed the changes of software prices during the period 1984 to 2000 and concluded that the average annual growth rates of quality-adjusted prices of personal computer operating systems were negative, ranging from -15% to -18%. Similarly, those for productivity suites generally ranged between -13% and -16%. Moreover, the price declines were generally greater in the latter half of the samples. (It may be that productivity-boosting innovation in the sector provides one explanation to reconcile such dramatic declines in prices of software with the strong data on the profitability for the sector.)

---

21 The sample included data for personal computer operating systems and productivity suites advertised in PC World magazine by retail vendors during the time period 1984 to 2000.
Table 3. Top Software Companies – Revenue Growth Rates

(annual change 2006-2007, %)

<table>
<thead>
<tr>
<th></th>
<th>Top 500 firms</th>
<th>Top 50 firms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corporate revenue growth rate</td>
<td>23.5%</td>
<td>16.6%</td>
</tr>
<tr>
<td>Of which, software and software-related services</td>
<td>23.3%</td>
<td>16.2%</td>
</tr>
</tbody>
</table>


Box 4 – Revenue of the Top 500 Software Companies

A recent survey by Software Magazine (2008) provides some insight into the economic scale of the software sector, presenting data for the top 500 software firms (out of the many thousands in the global economy). The data indicate that in 2007 the top 500 software companies as a group reported aggregate revenues of some USD 750 billion, with over one-half attributable directly to software and software-related services.

Revenue of the Top 500 Software Companies, 2007

<table>
<thead>
<tr>
<th></th>
<th>Total for this group of firms</th>
<th>Average firm in this group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corporate revenue (in USD)</td>
<td>750 bn</td>
<td>1.5 bn</td>
</tr>
<tr>
<td>Of which, software and services revenue (in USD)</td>
<td>394 bn</td>
<td>0.8 bn</td>
</tr>
<tr>
<td>Share of software services in total revenue (%)</td>
<td>52.50%</td>
<td>52.50%</td>
</tr>
</tbody>
</table>

Note: bn = billions of US dollars.

Indicators based on unit counts

40. A general problem with valuation approaches to software statistics is that they may fail to take into account certain types of software investment or use (as discussed Box 3). Therefore, as a complement to valuation-based indicators, analysts sometimes employ unit count approaches to monitor software developments. These approaches may rely on surveys or distribution data to track the distribution, installation or use of various types of software. In some cases, they rely on automated Interned-based methods. Such approaches are useful, for example, in providing information on activity related to some types of software distributed via non-traditional channels and that might not be reflected in standard economic statistics on the sector (e.g. software that is distributed free of direct charges to the user).  

22 For an example of such data on applications employed in enterprises, see Saugatuck Technology (2007).
Although not completely free from biases, they provide a useful additional set of information for use in assessing the size and evolution of various software markets.23

41. Market penetration rates provide one example of an indicator based on such approaches. This refers to the proportion of a total available market that is serviced by a given product or product type. In the case of software, market penetration rates can be presented as the ratio of the number of users or installed units of a specific software product or product category to the total size of the market. There are no comprehensive studies that address the issue of global market penetration rates. Rather, the existing studies generally assess market penetration rates of particular software products in selected countries or selected types of software globally.

42. The Netcraft Web Server Survey provides an example of a unit-count approach based on an automated Internet-based survey. This survey has tracked the growth in the number of Internet sites, highlighting its increase since 1995 (Figure 5). The survey identifies the software used on the various sites in order to provide up-to-date information on market penetration rates (Figure 6). The unit counts underscore the large size of the market for the leading server software products (e.g. 87 million servers using Apache, 62 million using Microsoft IIS). (A review of studies on OSS market penetration rates is presented in the Annex 1 at the end of this paper, covering both survey and automated internet-based exploration methodologies.)

Software and the labour market

43. ICTs, including software, have become a major source of employment creation across the OECD countries as well as a number of leading developing countries. Moreover, in view of the nature of software as an intangible product based on intellectual processes, the sector is particularly dependent on a highly skilled labour force. According to UNCTAD (2002), “The computer software and services industry is a key example of knowledge production, as the value of what a software company produces is almost entirely in the knowledge embodied in its products and services. It is a fast growing industry producing high value services for its customers.”

44. Employment in the ICT sector has experienced growth in most OECD countries for the last decade. Figure 7 presents data from 27 OECD countries on employment in the ICT sector in 1995 and 2004. In all the countries except Portugal, the share of ICT specialists in total employment increased during this period, in some cases substantially. As of 2004, six of the countries reported ICT-related employment shares of greater than 4%.

45. More specifically, and for a more recent period, data on the share of computer-related employment in total employment are available for the EU15 and the United States. This indicator ranges

---

23 Survey-based results on market penetration are not free from certain risks and biases, particularly where they cannot be automated and depend on knowledgeable respondents. They may fail to take into account the intensity of use by respondents with respect to the various software products (e.g. in cases where there are competing products held by an individual user). In cases where surveys focus on distribution, the availability of free software may lead to wrong impressions in that the number of downloads may be much greater than the number of actual users. In addition, some of the surveys are based on relatively small samples of firms or users and face sample-selection problems. Another issue related to user surveys concerns the potential respondent bias. According to the results of the FLOSSPOLS survey, software users are very often unaware of the openness of software they use. In fact 29.9% of OSS users claimed not to use “open source software” but reported they use “Linux”, “Apache” or other open source software products. Such lack of awareness could translate into biased survey results. (The FLOSSPOLS interviews were made in 2004 among employees of 955 public sector organizations in 13 EU countries. The results are available at http://flosspols.org.)
from about 0.5% to 3.5% for the EU countries during 2002 and 2006 (Figure 8). For the United States, a similar measure corresponded to around 2.4% of total employment between in 2004. As the US data are more detailed, software specific occupations can also be identified; the data show that computer programmers and computer software engineers accounted for around 1% of total employment in the same year.

46. Overall there has been a tendency for growth in ICT employment, albeit with some variation by country in recent years. This has been accompanied by increased demand for highly skilled workers to meet the demands of new and changing trends for particular technical skill sets. For example, in the United States as of 2006, the Bureau of Labor Statistics (BLS, 2008) estimated employment in just one software related category – computer software engineers – to be about 860 000. Through 2016, BLS projects this number may increase by 38%.

47. The demand for highly skilled labour has its origin in the nature of software innovation, which is not a physical capital-intensive activity but rather a creative human capital-intensive one. A software developer needs to be able to understand various aspects of a given (often complex) task and to address them through software modelling. In doing so, a software developer needs to display the ability to deal simultaneously with numerous details of a software task and, to possess well-developed abstract thinking skills. Software developers work mostly in teams. Software teams display high levels of team learning, constructive software practice patterns, excellent communication, appropriate role division and good integration of expertise. Even though some software teams are associated with project work in conventional software development firms, they may also be geographically distributed and supported by co-operative work systems. They can also be self-organising, facilitated by the Internet, and motivated by non-commercial goals, as in open source communities. The importance of human capital input in the software sector was underscored in replies to the OECD business questionnaire (Box 1 and Table 5), where trained human capital was selected as the most important feature for successful innovation activities (OECD, 2008a).

48. In view of the large and growing employment shares related to software, OECD countries are being challenged to educate increasing numbers of graduates with skills appropriate for employment in the ICT area. Enterprises as well are being challenged in recruiting and retaining talent, which is a very important issue for firms looking to remain competitive in terms of innovative software development. Clearly, the human capital dimension is set to play an important role in relation to software innovation.

24 During this period, the change in employment in the sector was mixed across the EU15 countries with some seeing a decline in the share (e.g. France, Ireland and Austria). A portion of these declines could potentially be related to the reported increase in the offshoring of ICT-related activities. (For further OECD research on the potential offshoring of ICT-intensive occupations see van Welsum and Vickery (2005) and van Welsum and Reif (2006a, b)). Another explanation for diverging employment trends may relate to differences in the rate of adoption and integration of new technologies including the automation or digitisation of tasks.

25 In an article entitled “Reaching for the Stars: Who Pays for Talent in Innovative Industries” Andersson et al. (2008) present the following conclusion: “software firms that operate in software sectors with highly skewed returns to innovation, or high upside gains to innovation, are more likely to attract and pay for highly talented workers. Such firms do so first by paying more up-front in starting salaries to attract skilled employees and second by rewarding workers handsomely for experience or loyalty.”
**Figure 5.** Evolution of the Total Count of Internet Sites, All Domains, August 1995 to July 2008


**Figure 6.** Top Developers, Web Servers, Shares Based on Unit Counts (in %), July 2008

Note: Shares based on unit counts from 175,480,931 identified sites.

Notes:

1) ICT specialists, who have the ability to develop, operate and maintain ICT systems. ICTs constitute the main part of their job.
2) In some cases, the data may not be strictly comparable: i) across time due to changes in definition and ii) across countries, as data classifications for EC and non-EC member countries have not been harmonised.

Figure 8. The share of computer-related employment in total employment, EU15, 2002 and 2006; USA, 2004

Note: For the EU countries, the ISCO88 categories “213 computing professionals” and “312 computer associate professionals” were used as a proxy. For the US, the CPS categories “110 computer and information systems managers”, “1000 computer scientists and systems analysts”, “1010 computer programmers”, “1020 computer software engineers”, “1040 computer support specialists”, “1060 database administrators”, “1100 network and computer systems administrators”, and “1110 network systems and data communications analysts” are used.

Source: Authors’ calculations based on EULFS and US Current Population Survey (CPS).

Software innovation – key features

Concept of software innovation

49. Innovation may be broadly defined as the successful commercial introduction of a new product or process. More specifically, according to the OECD’s Oslo Manual (2005), innovation refers to “implemented technologically-new products and processes and significant technological improvements in products and processes. An innovation has been implemented if it has been introduced on the market (product innovation) or used within a production process (process innovation). Innovation involves a series of scientific, technological, organisational, financial and commercial activities.” In this context, it is important to note that research and development (R&D) activity is just one element – albeit an important one – in the process of software innovation.

50. Software innovation can be seen as a process leading to:

- development of a novel aspect, feature or application of an existing software product or process; or
- introduction of a new software product or process or an improvement in the previous generation of the software product or process; and
- entry to the market or use within the production process.

(Box 5, below, provides an illustrative list of innovations on the horizon in software.)

51. Incentives are the fundament for purposeful activities including those that lead to software innovation. Direct economic incentives can be provided through earning economic rents, and there are various sources of rent for an innovative software developer. Most of them refer to the profits earned, but
can also include lower costs of production or increased efficiency of production and service delivery. Some of the economic rents are directly associated with the product (e.g. profits earned through sale of the software product or through sales of hardware with embedded software), others are linked to revenues from activities, such as services or advertisements that are provided by the software developer together with software product. With the advent of such developments as cloud computing and collaborative approaches, the possibilities for finding revenue “sweet spots” are becoming more diverse.

52. Beyond the direct monetary incentives, there are also quasi-non-monetary incentives for software innovation. While these may not yield immediate economic returns, they often offer some long-term economic advantage. In the case of an enterprise, examples include enhancing the firm’s reputation or market share, testing a new product concept, or exploration of new markets. In the case of individuals, this may include gaining experience and education, fame, signalling of skills or ideological motives (e.g. community identification or altruism), among others.

<table>
<thead>
<tr>
<th>Box 5 – Examples of software innovation on the horizon?</th>
</tr>
</thead>
<tbody>
<tr>
<td>The software press and Internet media offer hints of a wide range of innovation just over the horizon. A few illustrative examples are listed below:</td>
</tr>
</tbody>
</table>

Business applications – The Enterprise 2.0 Conference (http://www.enterprise2conf.com/) sees innovation as emerging to deliver a set of “technologies and business practices that liberate the workforce from the constraints of legacy communication and productivity tools like email. It provides business managers with access to the right information at the right time through a web of inter-connected applications, services and devices. Enterprise 2.0 makes accessible the collective intelligence of many, translating to a huge competitive advantage in the form of increased innovation, productivity and agility.”

User interface – Craig Mundie, Chief Research and Strategy Officer for Microsoft Corporation, argues that one of the next big innovations in computing will involve user interface and accessibility (BBC Interview, 16 May 2008, http://www.bbc.co.uk/worldservice/specials/924_interview_archiv/page12.shtml). He foresees change in the next 5 to 10 years allowing the computer to interact with users more in the manner of a human being, “to allow many more people to have facile use of computing” through language advancement (e.g. spoken commands, use of multiple languages and increased computer vision). He notes that some on board computers in cars use spoken-command inputs, constituting an early example of this type of innovation.

Network-user interface enhancement – Further evolution of these interfaces may result in more standardisation, with the Internet and operating systems more closely integrated. As one source puts it, “All the major developers of operating systems for PCs, network computers, and workstations, including Apple, Microsoft, IBM, and Sun, are hard at work integrating […]” (EMC Publishing, http://www.emcp.com/intro_pc/reading4.html).

Enterprise Application Integration (EAI) – Gartner (2001) sees EAI as “unrestricted sharing of data and business processes among any connected application or data sources in the enterprise.” Enterprise networks will permit better connection of applications, allowing for linkages between internal IT departments, distributors, suppliers, subsidiaries, partners, and clients. According to Wintergreen Research (Wintergreen, 2006), companies that achieve faster time to market through EAI can achieve competitive advantage.

---

26 Recent studies (e.g. Maurer and Scotchmer, 2006; Bitzer et al., 2004; Feller et al., 2005; MERIT, 2006) have identified a variety of such motives that provide incentives for developers of such products.
Patterns of software innovation

Basic analytical framework

53. The microeconomic literature often employs a simple, basic framework as a starting point for systematic analysis of innovation mechanisms in any sector. This framework conceptualises the innovation process as a linear sequence of events; it starts from the premise that innovators can be viewed as economically rational agents, motivated by a specific set of goals. In striving to attain those goals the innovators undertake a process beginning with their recognition of a potential innovation, followed by scoping of the idea, research, prototyping, testing, introduction of the new product to the market and full commercialization (Figure 9):

![Figure 9. Software innovation – stylized sequence of events](image)

54. At the R&D stage, a potential innovator launches purposeful R&D activity in order to create a product with particular goals in mind. Research time is costly either in monetary terms (e.g. wages, monetary cost of investments) or in non-monetary terms (e.g. time and effort devoted to R&D). Some of the determinants that affect the cost of research include the availability of human capital, the stock of public knowledge (e.g. resulting from research conducted in the public sector or from strong business-academic links), access to foreign inventions and to previous developments in the industry, and a sound financial market together with availability of direct public financial support for business R&D (Jaumotte and Pain, 2005a, b, c, d).

55. In the economy generally, innovation tends to be associated with purposeful R&D activity whose intensity depends in turn on economy-wide conditions and specific science-related policies and institutions (Fagerberg et al. 2004, Jaumotte and Pain, 2005a, b, c, d). However, specific cases of innovation may not have a proportionate relationship to research; efforts may focus on expansion and recombination of existing findings rather than new discoveries. In some cases, innovation (in the sense of successful commercial introduction of a new product or process) may take place without requiring extensive R&D.

56. In the case of software, the process of R&D can either start from scratch or build on previous innovations, often involving some degree of collaboration. R&D in the sector is often cumulative, with existing programming “solutions” forming the starting point for further developments. Thus, software development often takes the form of a multi-stage process that builds on previous innovative steps. As a result, releases of new products and product enhancements may take place relatively frequently.

---

27 For a general introduction to this perspective, see Mas-Colell, Whinston, and Green (1995), Kreps (1990) and Varian (1992). For more specific modelling techniques, see Grossman and Helpman (1991a, b) and Aghion and Howitt (1992).

28 E.g. in the United States, the U.S. Government is the principal funder of basic research, while industry funds substantial portions of applied research and product development.

29 For more information on R&D strategies, see chapter 2.
57. A product that is ready for commercialization moves into the market entry phase. The outcome of this phase is uncertain. Products may fail to navigate the various barriers to entry and competitive pressures. Naturally, this is a critical moment in the innovation process. Commercialized innovation is in many ways the successful exploitation of new ideas. In a competitive market environment, when the R&D process delivers the first-best solution to a consumer need, it may seem logical that successful commercialization should follow. A rational consumer might be assumed to choose the highest quality product cæteris paribus. Thus, in the case of cumulative innovation in the software sector, a significant product enhancement might tend to be followed by successful introduction to the market. However, the existing economic literature points to several known general market features that could influence the likelihood of a successful commercialization of new innovations. There are at least three main groups of such features including product quality, the degree of competition and network effects. Market presence is critical to attainment of the objectives of most software developers. Generally, there are two types of developer objectives related to the incentives for innovation: monetary (maximization of monetary profits during market presence) and quasi-non monetary.

58. As for market exit, a long expected product life extends the duration of the period over which the objectives of a developer are achieved and sustained. However, compared to products in other industries, the length of life cycles for specific versions of software products may be rather short (FTC 2003). While this depends in part on displacement by new generations of software (Harter et al. 2000), it is also affected by the quality of IPR protection (Jaumotte and Pain, 2005a). Weak legal standards or poor IPR enforcement result in high rates of digital piracy that can, in turn, significantly reduce potential profits for software developers (BSA, 2007). As Feller et al. (2007) note, a sound IPR regime is a necessary protection for innovators from unwanted expropriation of their creations by third parties.

59. This conceptualisation presents a simplification of reality in the software sector. In practice, software developers may engage in collaborative approaches at any point in the process (discussed in the next section) and may employ a variety of business models to achieve objectives or extract value. In some cases, for example, developers may focus on their comparative advantage in delivering just a part of the process (e.g. contributing testing services or marketing and distribution expertise in a collaborative project); in others they may take the lead throughout a project. Larger firms may employ different approaches in different projects that are running simultaneously.

Collaboration and open innovation

60. Collaborative and “open” approaches may be used to enhance the basic innovation process in cases when a software developer wishes to leverage innovation efforts by bringing external knowledge or capacities into the software development process. Generally there are three pillars of open innovation: (i) sourcing and integrating external knowledge from customers, suppliers, universities and research organisations, or even competitors; (ii) bringing ideas to market (not necessarily finished products); selling, licensing, trading intellectual property as part of strategies to multiply technologies; and (iii) working in alliances in order to capitalise on complementarities (Chesbrough, 2003; Enkel and Gassmann, 2004; Chesbrough et al., 2006; OECD, 2008b).

---

Note: Network effects occur if the number of other customers who use a given product, matter for the value of that particular product to a potential customer (Shy 1996, 2001). In the software sector network effects occur when use of a particular program one individual indirectly increases the value of this program for other users. For example, an additional user of a particular messenger programme (e.g. Google Talk or Skype) increases indirectly its value for all other potential users, for whom the number of current clients is a component that affects the product’s value.
61. Although collaboration in software innovation initiatives takes a variety of forms, some common examples might include:

- engagement of individual PhD and postdoctoral researchers in innovation projects;
- partnerships or contract research with academia through framework agreements, often with a focus on long-term cooperation (including the creation of joint laboratories or high-tech zones by ICT firms (including SMEs) on university campuses);\(^{31}\)
- industrial technology alliances, R&D partnerships and consortia between ICT firms (e.g. for upstream research or for joint product development) or between ICT firms and partners or customers outside the sector; and
- prospecting for valuable new ideas from individuals, software communities and start-ups with promising research (including through venture capital, incubation and acquisitions).

62. The software industry works actively to capitalise on such external sources of knowledge. Major incentives include cost and risk reduction (especially for pre-competitive R&D) and improved possibilities to enter markets with jointly developed technologies (Freeman and Soete, 2007).\(^{32}\) These collaborations are increasingly international, spanning various segments of the ICT sector and adjacent industries (e.g. biotechnology).

63. The use of collaborative and open approaches has been strengthened by globalisation and technological progress. Internationalisation is facilitated by the increasing use of ICTs as a basis for the science and technology infrastructure (e.g. broadband-linked research networks) and by the fact that international research collaborations have been encouraged by policy programmes (e.g. the EU FP7’s focus on research cooperation with entities from Asian countries) and specialised organisations (e.g. the International Technology Roadmap for Semiconductors).\(^{33}\) Long-standing public research organisations with dedicated ICT foci (Fraunhofer in Germany, Battelle in the US, VTT in Finland, TNO in the Netherlands) are also increasingly engaging in global research alliances in order to leverage their efforts. Moreover, economic liberalisation has facilitated the establishment of international business ties among enterprises via trade and investment.

Software Innovation: Beyond the Software Sector

64. Software development takes place across the economy. The tight links and complementarity of the software sector with other industries are highlighted in several OECD studies (2002, 2004, 2006a and

\(^{31}\) Examples of partnerships with universities include: Oracle and CERN (European Organisation for Nuclear Research) for grid computing technologies; Microsoft, Nokia, Hitachi, and Toshiba with research centres at the University of Cambridge; Fujitsu collaborating with the Universities of Tokyo and Cambridge on quantum technologies.

\(^{32}\) Costs for research conducted under such approaches can be comparatively low in cases where some of the participants are motivated by incentives other than direct monetary interest in a specific commercial product. This may take place for example when one of the parties has the expectation of eventually capitalising on the experience in developing a commercial product at a later stage (Iansiti and Richards, 2006).

\(^{33}\) The objective of the International Technology Roadmap for Semiconductors (ITRS) is to ensure advancements in the performance of integrated circuits and remove roadblocks to the continuation of Moore's Law. This assessment, called roadmapping, is a cooperative effort of global industry manufacturers and suppliers, government organizations, consortia, and universities. More information can be found on the ITRS website at: http://www.itrs.net/ (last accessed on 27.09.2008).
b). These studies point to the relationship of the software sector to information, computer and communications industries through technological links and through the business activities of firms that are often present in multiple markets. Moreover, a substantial volume of software is produced outside the traditionally defined “software industry” (e.g. van Genuchten, 2008), as is often the case for software developed in-house or for products in sectors such as consumer electronics or cars.

65. Data on R&D expenditures related to software underscore the breadth of such investment across the economy, a point confirmed in several national R&D surveys. According to Young (1996), for example, at the beginning of the 1990s, about 40% of services firms in Japan and Italy undertook some form of IT research activities including software development. In Denmark, about 75% of all R&D investment reported by “other services industries” was computer-related. In Canada, over half of all R&D in the services industries was software-related.

66. In-house needs constitute a major driver of software development in firms beyond the sector. Even though the exact quantification of software remains a major challenge, existing estimates suggest that software developed for own needs ranges from 20% to 40% of software production. A further 40% to 50% of the market is in custom software, produced for users by software service providers (Parker and Grimm, 2000; Grimm et al., 2002).

Technical aspects of Software Innovation

67. Software innovation occurs in an abstract and digital space, which makes it technically different from some other industrial disciplines such as electrical or mechanical engineering. Whereas other areas may be subject to numerous physical constraints, software development, by contrast, is physically constrained only by hardware for which it is designed. Within this relatively unconstrained environment, the software development process strives to optimise across a number of dimensions in what can be a highly complex task.

68. Development of a high quality software product may require simultaneous consideration of such characteristics as functionality, reliability, usability, efficiency, maintainability and portability, among others (ISO, 2001). The software sector has responded to this complexity in two general ways. On the one hand, there has been an effort to advance the field in a rigorous manner based on engineering disciplines. On the other hand, software innovators have aimed to create certain frameworks that enable less advanced users to apply their ideas in developing software to meet their specific needs. These two concepts are discussed below.

Software Engineering and Development Tools

69. Software engineering aims to apply engineering approaches to software development, deployment and maintenance. It is a relatively new field, with the term “software engineering” first established during a conference of the NATO Science Committee held in 1968 (NATO, 1969). SWEBOK, the IEEE (2004) Computer Society guide to the field, defines software engineering in terms of “the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software...”. The guide lists 10 knowledge areas covered by the field including: software
requirements\textsuperscript{37}, design, construction, testing, maintenance, configuration management, and quality, plus software engineering management, process, and tools and methods.

70. Software engineering tools consist of programs or applications that assist in creating, configuring, maintaining, testing or supporting software-intensive systems. These tools evolve continually in response to the development of new methodologies for creating software systems. Over the years, they have become quite sophisticated and now cover the full lifecycle of software systems. They are often bundled together to create an Integrated Development Environment (IDE), offering access to toolkits for software development in one or more programming languages. Illustrative examples of IDEs include Eclipse (an open source solution), JetBrains’ IntelliJ IDEA (a Java IDE) and Microsoft Visual Studio (a Windows IDE).

71. Progress in software engineering and development tools has been driven or facilitated by a number of factors including, for example:

- The exponentially growing demand for software systems, which has increased in terms of both the number of systems and the complexity of the individual systems. Moreover, software development is very skill-, and labour-, intensive. Consequently, participants in the software sector have recognised that new tools, methodologies and processes are required to enhance their productivity, satisfy quality requirements and fulfil demand.\textsuperscript{38}

- \textit{Advancement in hardware technologies, including processor speed}\textsuperscript{39}, display technology, storage capability and networking (Taylor and Van der Hoek, 2007). For example, increases in processor speed have boosted the analytical potential of tools and high-speed network communications have broadened the participation of stakeholders in the software design process.

- \textit{The need for a degree of consistency in a changing environment and the push for technological convergence.} The extremely complicated, heterogeneous and dynamic situation in the software sector is challenging software developers to take into account of multiple environmental aspects and contexts while delivering consistency of software design decisions and multidimensional access to design data (Taylor and Van der Hoek, 2007). Advanced approaches and tools are needed to tackle some of these challenges.

- \textit{Globalisation, enhanced power and diffusion of ICT infrastructure, and the tools supporting collaborative work} have enabled software engineers to engage expanded resources around the world for purposes of research and development.

72. In sum, advances in networks and computing power have resulted in an explosive growth in sophisticated applications and services that touch almost all aspects of society. Software engineering is playing a central role in this process, both contributing to these advances and helping to respond to the demand for further progress.

\textsuperscript{37} According to Donzelli \textit{et al} (2006), requirements engineering (concerned with methods, techniques and tools for eliciting, modelling and analysing software requirements) is one of the most critical areas of software development. They note that requirements errors are among the most costly and time-consuming to correct and that erroneous or omitted requirements are often indicated as the main reasons for project failures.

\textsuperscript{38} Examples of efforts to enhance productivity include the use of Model Driven Development, agile processes and SaaS.

\textsuperscript{39} Also, the development of parallel processing is contributing to expand the possibilities for software.
Software frameworks

73. The appearance of software frameworks during the 1990s marked a new approach to software development, particularly with respect to applications. A software framework is "the skeleton of an application that can be customized by an application developer" (Johnson, 1997). Software frameworks can include support programs, code libraries, a scripting language or other software to help develop and assemble the different components of a software project. Frameworks facilitate software development by allowing designers and programmers to spend more time on delivering functionality and less time on the more tedious low level details and routine aspects of providing a working system. Thus, a high-quality framework increases developer’s productivity, in particular for those involved in big and complex projects. A related approach is the use of Service Oriented Architecture (SOA) in order to promote efficient development of software systems (Box 6).

74. In the construction of frameworks, two types of elements can be distinguished (Pree and Sikora, 1997). The rigid and immobile elements (“frozen spots”) of frameworks provide the necessary structure and cannot be changed by a developer; they are the fundamental and basic elements of a framework system that shapes its overall architecture. Programmers focus on the flexible elements (“hot spots”) in the framework, which they can modify in order to meet a given project’s functionality requirements. There are numerous fields of application of software frameworks including: graphical editors, music programmes, programming software, multimedia and web applications, among others. Examples of software frameworks include Java Native Interface, Salesforce.com’s Force.com, .Net Framework 3.5, and Apache Struts.

---

40 The Techweb Network offers the following explanation of the term application framework, “A set of common software routines that provides a foundation structure for developing an application. Frameworks take the tedium out of writing all the program code for an application from scratch. Object-oriented application frameworks, which are the norm today, are structured as a class library.” The full explanation is available at: http://www.techweb.com/encyclopedia/defineterm.jhtml?term=applicationframework (last accessed 01.08.2008).

41 In some countries (e.g. in Spain), governments have supported software development frameworks with a view to promoting openness.

42 Illustrative examples include: 1) Sun’s Java Native Interface http://java.sun.com/j2se/1.3/docs/guide/jni/, 2) Microsoft’s .NET Framework http://www.microsoft.com/downloads/details.aspx?familyid=333325FD-AE52-4E35-B531-508D977D32A6&displaylang=en, 3) Salesforce.com’s Application Framework, which the firm claims will deliver “the ability to customize existing applications or build applications from scratch without writing any code.” http://wiki.apexdevnet.com/index.php/Application_Framework, 4) Apache claims that its Struts framework aims “to leverage existing standards by producing the missing pieces we need to create enterprise-grade applications that are easy to maintain over time. […] Struts 1 is recognized as the most popular web application framework for Java.” http://struts.apache.org/. (All websites cited, last accessed 03.08.2008).
Business Models and Software Innovation

75. A business model might be considered as “the translation of strategic issues, such as strategic positioning and strategic goals into a conceptual model that explicitly states how the business functions. The business model serves as a building plan that allows designing and realizing the business structure and systems that constitute the company’s operational and physical form” (Osterwalder et al, 2005). With technologies evolving rapidly in the software sector, strategic issues continue to shift and new ways of doing business are emerging. In many cases, firms are finding it necessary to re-examine their business model and find alternative ways of raising revenue. The interaction of business model and technological developments can stimulate – or be stimulated – by innovation.

76. During the 1950s and 1960s, IBM and other major providers frequently employed a revenue development strategy by which their software was provided free of charge or bundled with proprietary hardware with the goal of increasing revenue from the sale of hardware (Campbell-Kelly and Garcia-Swartz, 2008). As demand for software grew in complexity, hardware producers (notably IBM) shifted to more open design approaches for hardware that enabled external software developers to deliver innovative software and as more hardware became accessible to benefit from significant economies of scale (van Genuchten, 2007).

77. Nowadays a broad range of business strategies and practices are employed by various commercial and non-commercial entities in the development and distribution of software products and services. No single development strategy or business model has gained primacy. Rather, the software sector is characterized by its dynamism, with a shifting range of commercial strategies and practices and a diversity of market participants. The proliferation of models is largely the result of new platforms and distribution channels, in particular with ever increasing broadband connectivity.

78. One useful approach to analysing business models for software firms is provided by Rajala et al (2003). Noting that each company and business model is constrained by external variables such as the macroeconomic framework conditions, financing conditions and the competitive environment, these authors decompose a generalised business model into four elements (Figure 10):

- Product strategy: describes the core product and service proposition of a software business and the way the product development work is organised.
- Revenue logic: describes the sources of revenue and the way the software vendor generates revenue from these sources.

Box 6 – Service Oriented Architecture (SOA)

Under service oriented architecture (SOA), enterprises and other users can structure their computer systems in modules that can be accessed in various combinations and by various users as necessary to deliver services. SOA, through its emphasis on modular design, permits interoperation of numerous services, where access to these services can be executed autonomously. This is achieved by imposing a certain structure on the basic applications and introducing a set of smaller flexible modules that can be created and modified according to current demands (Bell, 2006). According to a report of WinterGreen Research (2006), this segment of the market is set to grow rapidly over the next few years. The authors state, “SOA markets at $450 million in 2005 are expected to reach $18.4 billion by 2012.”

Parker and Van Alstyne (2008) provide a useful assessment of the economic context for platforms.
- Distribution model: describes the way marketing and sales of the product and service offering have been organised and identifies the sellers and marketers of the product and service offering.

- Service and implementation model: describes how the product offering will be dispatched to the customers and deployed as a working solution, including the set of services and actors implementing them.

Figure 10. Elements of a business model


79. There are many alternative ways of classifying business models, although most of these can be brought back to the four main elements described above. Table 4 gives one example of a list posted on a discussion forum on the theme “Trying to list all possible software business models”. Certainly, many business models exist, employing various approaches to each of the four elements. These approaches can be operationalised via a multitude of business policy levers concerning such issues as openness of innovation processes and strategic alliances, development of direct and indirect revenue sources, and approaches to product licensing, among many others.

---

The discussion forum can be found here: http://www.linkedin.com/answers/technology/enterprise-software/TCH_ENT/47038-3420784?browseldx=0&sik=1183416866316&goback=amq (last accessed 07.05.2008).
A review of information on business models collected by the OECD Secretariat from public sources (Box 7) confirmed that individual firms employ different mixes in different contexts as they pursue innovation and growth strategies. The following examples drawn from these sources illustrate just a portion of the diversity of approaches applied within these firms and across the sector:

- CA Inc. uses so-called “flexible licensing” as part of its business strategy.
- Electronic Arts sells interactive games with advertising both in their games and on their web site.
- Lawson relies on a combined approach of software licence fees, consulting, and training and implementation services.
- Microsoft applies different models in different contexts, in some cases using traditional models based on direct licensing and sales, often pursuing partnerships, sometimes aiming for indirect revenue streams (e.g. via embedded advertising on web sites), and employing various degrees of openness, among other approaches.
- Misys specifically engages in partnerships and collaborations to innovate and capture new market opportunities.
- Hitachi and Oracle specifically engage in mergers and acquisitions.
- Sun Microsystems focuses on sales of hardware products, and simultaneously supplies software based mostly on the open-source licensing principle.
- Atari has a global strategy of local presence to adapt their products to the local market; they have specifically built up a broad-based catalogue in order not to rely on a single product; and they combine franchises and licences.
- Novell and Linux are two examples of hybrid structures, combining open innovation and other business model elements.
- Mozilla is an example of a foundation seeking to promote software innovation through an open framework.
### Table 4. Alternative types of software business models

<table>
<thead>
<tr>
<th>Consumer models</th>
<th>Business software models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proprietary Software</td>
<td>Open source</td>
</tr>
<tr>
<td>Public Domain</td>
<td>Public Domain</td>
</tr>
<tr>
<td>Freeware</td>
<td>Bundled with hardware (e.g. HP Openview)</td>
</tr>
<tr>
<td>Shareware</td>
<td>Proprietary Software</td>
</tr>
<tr>
<td>Adware</td>
<td>Consulting ware (e.g. E&amp;Y’s tax readiness software - you get it when you sign the audit contract)</td>
</tr>
<tr>
<td>Demoware</td>
<td>By the seat</td>
</tr>
<tr>
<td>Trialware</td>
<td>By the named user</td>
</tr>
<tr>
<td>Hardware supported (ships with hardware)</td>
<td>By concurrent user</td>
</tr>
<tr>
<td>Pluginware (available if you buy the main application or OS)</td>
<td>By average # of users</td>
</tr>
<tr>
<td>Premiumware (basic version is free, added functions require $)</td>
<td>By location</td>
</tr>
<tr>
<td>Subscription ware (e.g. anti-virals)</td>
<td>By server</td>
</tr>
<tr>
<td>Upgrade ware (e.g. tax software - cheaper if you bought last year)</td>
<td>By CPU</td>
</tr>
<tr>
<td>Permanent ware (buy once and all the future versions are free)</td>
<td>By CPU class</td>
</tr>
<tr>
<td>Donationware (donate to a charity and it is free)</td>
<td>By transaction</td>
</tr>
<tr>
<td>Bundleware (2nd and 3rd apps in a bundle)</td>
<td>By revenue</td>
</tr>
<tr>
<td>...and probably 1 000 more</td>
<td>By employee count</td>
</tr>
</tbody>
</table>

Source: Discussion forum on Linkedin, post by Doug Houseman, Associate Director at Capgemini, 19 May 2007.

---

81. The case of embedded software merits special attention in that the business models involving embedded software generally create value by selling the hardware product to customers, whereby the software contributes indirectly to the bottom line. For example, the software may add functionality valued by the consumer or permit more economical production for the producer (e.g. using a software switch instead of a hardware switch). There are also software developers that create value from such software directly, by selling embedded software products, as such, to hardware producers or integrators that incorporate the software in their products for subsequent resale as part of a hardware product.

### Approaches to Licensing

82. Approaches to licensing vary across a spectrum from closed-source licensing and open-source licensing. At one end of the spectrum, closed-source licensing focuses on innovation through the use of proprietary licenses and intellectual property protection. The idea is that keeping the source code closed, and relying on strong patent and copyright protection provide an incentive for software development by providing a means of appropriating monetary rewards for innovative thinking. Proprietary models keep source codes as property of the company and have tended not to allow for open-access and modification or alternative application of the code.

---

45 Here it should be noted that some open source developers use intellectual property protection as a means of specifying the permitted uses of the open software. For example, intellectual property can also be used to ensure openness in subsequent applications of the code.
83. Firms using proprietary models have a variety of channels for fresh innovative inputs. Some firms diversify by acquisition of firms that have innovated using hybrid models. User-driven development has also often been an important aspect of the innovation strategy in these business models. Moreover, firms using proprietary approaches may make available to outside developers information required for interoperability (e.g. API information). For platform developers, this can be essential in ensuring adoption, while for other types of software (e.g. application and middleware) interoperability can help to promote development of follow on innovation that adds further value to the original product. Ecosystems can develop around such software.

84. The other end of the spectrum is occupied by a range of open-source licensing models. The premise of these models is that innovation occurs when multiple developers are able to access source code in order to further the functionality of a program. Developers cooperate under a model of rigorous peer-review and take advantage of knowledge, talent and skills globally, which has proven conducive to innovation. A firm allows users access to its intellectual property, and in return it receives input in the development of tools that it would otherwise have had to come up with on its own. Open source licences support this strategy in two ways (Hope, 2003): (i) users cannot become co-developers of a tool unless they have access to that tool in a form that they can understand and modify, (ii) a requirement to provide access to source code in order to provide an incentive to contribute to a co-operative effort which would be hindered if potential contributors expect to be prevented from using the tool that they helped to create.46

### Box 7. Information from Public Sources on Software Business Models: Data Compilation Method

In conducting its preliminary assessment of business models in the software sector, the Secretariat built a small database drawing on public information. The primary sources included annual reporting by listed companies and public statements by various non-profit organisations; where available, this was based on reporting for 2007. The web site of the US Security and Exchange Commission (http://www.sec.gov/) proved to be a valuable source, as US-listed firms are required to submit periodic reports with certain standard information; these include both domestic and international software-oriented firms listed on the New York Stock Exchange (NYSE) as presented on 10-K and 20-F Annual Report forms. For privately held companies or international firms not listed on the NYSE, corporate websites were used to attain applicable information from annual reports and investment profiles. Additionally information from the NYSE European partner website Euronext (http://www.euronext.com) was used in a few instances. In order to compile information on open source foundations, the Secretariat directly accessed official foundation websites. For firms developing embedded software, information could be obtained from both the SEC website and corporate websites. Articles pertaining to collaboration between software-oriented firms, open source foundations and firms utilizing embedded software were taken from corporate and foundation websites directly. The data were compiled in a spreadsheet detailing information on business models, research and development, functionality and user input to innovation in order to draw comparisons related to their innovative activities. The exercise yielded information on innovation, interoperability, embedded software, open source, proprietary rights, collaboration, internal versus external development, and functionality, providing inputs for the subsequent analysis.

85. In practice, many software developers employ a combination of various licensing models. For example, Red Hat supplies proprietary software on top of its Linux offering. IBM offers integrated combinations of open source and proprietary software. Microsoft employs a wide range of business models and strategies, ranging from fully proprietary “closed source” offerings to “free ware” and fully open source offerings.47 In some cases, open-source foundations such as the Linux and Mozilla Foundations support the business models of open-source companies, providing intellectual property strategies, quality control and financial support for open-source initiatives. Some open source initiatives receive corporate sponsorship in addition to the revenue they may generate (e.g. from donations and philanthropy) in the

---

46 For more on open business models see: http://rsss.anu.edu.au/~janeth/OSBusMod.html, (last accessed: 07.08.2008).

47 E.g. Microsoft developed and actively uses two open source licenses approved by the Open Source Initiative – the Microsoft Public License and the Microsoft Reciprocal License – as well as a variety of other licensing schemes that allow users the right to redistribute or modify the original software.
course of their operations. It is very probable that such corporate contributions are made with strategic considerations in mind (e.g. development or promotion of a specific technology or encouragement of competition in a key area).

86. Open source software can serve as the basis for a variety of business models, although this is sometimes misunderstood. Even though open source can be “free” to developers and users, there are a variety of businesses built around this type of software. Examples of business models utilized in this segment of software include:

- **Professional Services** - Customization, Support, Training.
- **Dual License** - Sell a commercial version of the software to businesses that want to use it for commercial purposes and give away a free version to the open source community.
- **Enhanced Product** - Sell a commercial version that has substantially different features than the open source version for consumers.
- **Software as Service** – In some cases, give away the software code, but sell the software as a hosted web service either through advertising or subscription fees.
- **Bookware** - Give away the code, but sell the documentation.
- **Marketplace** - Give away the code, build a community marketplace, and take a cut of what all the people selling add-ons make.
- **Foundation** - Get corporate sponsorship and give away the code. (e.g. Apache)
- **Hybrid** - Blend several of these.

87. Across the spectrum of licensing approaches, collaboration is an increasingly important element of business models and innovation strategies for software firms. This collaboration can take place between firms that develop proprietary software or with open source organizations. There have been many initiatives to combine the benefits of open source and proprietary models to enhance innovation, as illustrated by the list of models presented above. The idea is always to take advantage of the resources available through open source and then add value that can generate revenue. The previous examples of collaboration are often achieved through the commercialization of open source software and sometimes through the use of patent agreements between open source and proprietary businesses or through a combination of open source and proprietary initiatives (Box 8). Interoperability becomes increasingly important in the specific context of such software business models, with mergers and acquisitions, collaborations, outsourcing, and offshoring.

<table>
<thead>
<tr>
<th>Box 8. Examples of “hybrid” models</th>
</tr>
</thead>
<tbody>
<tr>
<td>When an open source product is sold or commercialised with substantially different features this can often enhance the product, which is an important aspect of innovation within business models in terms of appealing to the end-user. It is also possible to provide the software code as open source while commercialising specific software applications as a hosted web service either through advertising or subscription fees (software as a service). Bookware is another alternative incorporating proprietary and open source initiatives in which the code is presented as open source but the documentation is commercialised. In a “marketplace” model the code is presented as open source on which a community marketplace is built and profit is made through the sale of add-ons, to both individuals and sponsoring businesses. Dual-licensing is another way that software companies can combine the benefits of proprietary and open-source initiatives through selling a commercial version of software to businesses, as well as giving a “free” version to</td>
</tr>
</tbody>
</table>
open source communities. In this way businesses can capitalize on their innovations while also contributing to open innovation. Other examples of collaboration come in the form of acquisition or creation of hybrid business units. Hybrid business models often combine elements from each of the examples mentioned above or focus on one primarily, but in general they are used to increase innovation through some form of collaboration. This is often an option for proprietary companies who wish to commercialise open source initiatives under their own legal structure.

The environment for innovation

88. The intensity of innovation activities is influenced by a variety of factors. Recent OECD research has highlighted certain of these, including the availability of human capital, the stock of public knowledge (e.g. resulting from research conducted in the public sector or from strong business-academic links), access to foreign inventions, a sound financial market together with availability of direct public financial support for business R&D, and the regime governing intellectual property rights (IPR, addressed in more detail in Box 9) (Jaumotte and Pain, 2005a, b, c, d).

89. The results from the OECD Business Questionnaire (Box 1 and OECD, 2008a) are helpful in identification of the importance of some of these environmental conditions as seen from a software developer business perspective, including across the four dimensions cited in Figure 10. The complete list of responses is outlined in Table 5, whereas the most important factors are highlighted in Figure 11.

90. The results from the questionnaire, which is illustrative and not necessarily representative, suggest that firms may tend to view availability of trained human capital as a crucial factor for software development. Customer requirements are also highly ranked with a particular impact on customer demands regarding quality costs and security. Other factors that were often underscored as highly important are: IPR protection; application of technological standards; legal, regulatory and administrative environment; customers’ requirements regarding interoperability and security issues. The remaining were considered of lesser importance, including customers financial strength and access to financing; technological infrastructure; language and culture; taxation and the physical distance to the customer.

Box 9. Software IPRs and innovation

Software-related innovations can be protected as intellectual property and, as is often the case with intellectual property in other fields of technology, remain vulnerable to imitation. To maintain incentives to innovate, governments have developed a variety of means to protect the rights of innovators in the software sector. International agreements, either bilateral or multi-lateral, provide a basic framework for the protection of IPRs in the software sector, including in some cases standards for minimum protection.

The specific mix of protections available varies among countries, but copyright protection is generally afforded to software innovators, as is some form of patent protection for software-implemented inventions. Other types of protection are also available to software innovators, such as protection for trade secrets or trademarks. A broad spectrum of software innovators has come to rely on IPR protection as an integral part of their business strategies; this includes most producers of both proprietary and open source software and firms or individual programmers operating under a wide range of business models.

Under current international arrangements, copyright protection for software innovations is quite broad and in theory available in countries representing the vast majority of the world economy. Members of the World Trade Organisation (WTO, with 151 member economies at the time of writing) are subject to the Agreement on Trade-Related Aspects of Intellectual Property Rights (TRIPS), which provides for computer programmes to be treated as literary works, eligible for protection under the Berne Convention for the Protection of Literary and Artistic Works (administered by the World Intellectual Property Organisation, WIPO). Under these accords, copyrights automatically come into effect with the

---

48 E.g., in the United States, the U.S. Government is the principal funder of basic research, while industry funds substantial portions of applied research and product development.
creation of the work and generally benefit from minimum standards of protection\textsuperscript{49} and national treatment in other signatory countries.\textsuperscript{50} Signatories commit to recognise and enforce this protection internationally.

Concerning patents, the WTO TRIPS Agreement states that patents shall be available in all fields of technology, a provision that applies in principle to software-implemented innovations. Other international agreements, including free trade agreements, are also relevant to the patentability of software. Yet, the detailed manner of implementation of patent protection for software-implemented innovations is not prescribed in full under these international agreements. Consequently, there is some variability in the processes for obtaining patent protection, and in the scope of patentable subject matter. In particular whereas in certain economies (\textit{e.g.} US) software is often protected with patents, there are economies (\textit{e.g.} European Union) where the existing legal regulations exclude software patentability.

Some observers consider patents and copyrights as complementary tools for the protection of intellectual property with respect to software (Einhorn, 1990): copyrights protect original computer programmes against unauthorised copying, whereas patents can be used to protect inventions (especially the underlying technical ideas and principles). Key differences arise in the characteristics of copyrights and patents: copyrights protect the work internationally without formalities, whereas patent protection is only granted in countries where the rights holder has applied for a patent (in certain economies patents cannot be applied to software, \textit{e.g.} European Patent Convention excludes patents on programs for computers as such). In practise, the duration of protection for a copyright is longer (generally amounting to 70 years or author’s life plus 70 years), whereas a patent usually gives the owner protection over 20 years.

\begin{table}
\centering
\begin{tabular}{|l|c|c|c|c|c|}
\hline
\multicolumn{2}{|c|}{Factor} & \multicolumn{4}{|c|}{Importance} \\
\hline
 & & High & Medium & Low & Not important & Unknown \\
\hline
Trained human capital & & 92\% & 8\% & 0\% & 0\% & 0\% \\
Customer requirements regarding quality & & 81\% & 15\% & 4\% & 0\% & 0\% \\
Customer requirements regarding costs & & 73\% & 19\% & 8\% & 0\% & 0\% \\
Customer requirements regarding security & & 73\% & 19\% & 8\% & 0\% & 0\% \\
Other * & & 71\% & 14\% & 0\% & 0\% & 14\% \\
Protection of intellectual property rights & & 68\% & 15\% & 8\% & 0\% & 4\% \\
Application of technological standards & & 67\% & 29\% & 4\% & 0\% & 0\% \\
Legal, regulatory and administrative environment & & 65\% & 15\% & 19\% & 0\% & 0\% \\
Customer requirements regarding interoperability & & 65\% & 31\% & 4\% & 0\% & 0\% \\
Security issues & & 63\% & 33\% & 4\% & 0\% & 0\% \\
Customer’s financial strength & & 40\% & 44\% & 8\% & 0\% & 4\% \\
Access to financing & & 33\% & 42\% & 21\% & 4\% & 8\% \\
Technological infrastructure and transport (\textit{e.g.} broadband coverage) & & 32\% & 44\% & 20\% & 4\% & 0\% \\
Language and culture & & 25\% & 46\% & 21\% & 8\% & 0\% \\
Taxation & & 27\% & 31\% & 19\% & 15\% & 8\% \\
Distance to customers & & 13\% & 50\% & 38\% & 0\% & 0\% \\
\hline
\end{tabular}
\caption{Perceived Importance of Environmental Factors to Software Developer Operations}
\end{table}

Notes: N = 26. The figure presents the share of firms identifying each factor as being of the indicated degree of importance to the firm’s software development operations.

Source: OECD (2008a).

\textsuperscript{49} E.g., the Berne Convention provides for a minimum term of protection.

\textsuperscript{50} National treatment in effect means that, for example, works that have a country of origin that is a Berne Union country benefit in all other Union countries from the same protection as the latter give to the works of their own nationals.
Figure 11. Factors of high importance for innovation, as assessed by software developers (% of replies, respondents could indicate multiple factors)

Notes: N = 26. The figure presents the share of firms identifying each factor as being of high importance to the firm’s software development operations.

Source: OECD (2008a).
ANNEX 1

REVIEW OF STUDIES ON MARKET PENETRATION RATES

This Annex outlines several studies on market penetration rates. The results are presented here in order to illustrate attempts to estimate the size of software market taking into account software that is not fully covered in the general statistics. In particular, many of these studies focus on open source software (OSS). This review is presented in three sections: 1) survey-based studies, 2) census-like studies, 3) country-focused studies.

1. General survey – based studies

Saugatuck Technology (2007) indicates that open source now accounts for roughly 10 percent of worldwide key enterprise on-premises software. The analysis projects that the OSS may expand its share to between 15 and 20 percent by 2010. The firm also expects explosive growth in mixed-source environments and that more than 35 percent of new commercial software implementations will include open source components by that year. The authors foresee that 4 or 5 master brands (some coming from a traditional software background) will influence or control at least 30 percent of the open source marketplace by 2012. Regarding other market forecasts, Gartner (2008) predicts that by 2012, more than 90 percent of enterprises will use open source in direct or embedded forms.

Concerning the market for operating systems, Gartner (2004) presented a study based on sales figures from computer vendors with pre-installed Linux. This study concluded that the Linux operating system was on 5% of personal computers sold worldwide in 2004. This, however, was not confirmed by the census – like studies by Net Applications (presented below).

Regarding office applications, there are a variety of sources referring to surveys and to the number of downloads published on the Open Office web site. According to Jupiter Research (2003) Open Office had a 6% market share among small and medium enterprises in 2003. For 2007, Freeform Dynamics (2007) estimated that market share of Open Office amounts to 7% for office use and 20% for home use. With respect to studies of a more narrow geographic scope, according to Méndez (2005) Open Office has 8.5% of the market among major North American companies, whereas in Germany according to TechConsult (2006) it is likely to be used by 8% of firms.

2. Census-like surveys

The best known of these market share indicators is the monthly Web Server Survey conducted by the research company Netcraft. It provides a “census” of accessible web servers on the Internet of which over a half run the OSS Apache web server application (Figure A1.1). However, in recent years Apache suffered a significant decline in share, which is related with the growing market share of Microsoft’s products and the introduction of Google's new product, GFE.

The measurement OSS is by nature difficult, given that the reuse and redistribution can occur without author’s authorization and given that a portion of OSS distribution occurs without any monetary payment involved.
Another automated census-like survey is provided by Timme (2004); this presents market penetration rates for mail servers. Timme performed a scan on existing mail servers in Europe and in the US based on IP addresses. The most popular programs are listed in Table A1.1. OSS servers accounted for almost half of the total market.
Table A1.1. Share of web servers

<table>
<thead>
<tr>
<th>Mail Server</th>
<th>Market share</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft ESMTP MAIL Service</td>
<td>23.89%</td>
</tr>
<tr>
<td>Sendmail *</td>
<td>22.38%</td>
</tr>
<tr>
<td>IMail</td>
<td>19.25%</td>
</tr>
<tr>
<td>Exim *</td>
<td>13.22%</td>
</tr>
<tr>
<td>Postfix *</td>
<td>5.57%</td>
</tr>
<tr>
<td>MailEnable</td>
<td>0.84%</td>
</tr>
<tr>
<td>Merak</td>
<td>0.29%</td>
</tr>
<tr>
<td>Mailer Daemon</td>
<td>0.24%</td>
</tr>
<tr>
<td>MERCUR</td>
<td>0.13%</td>
</tr>
<tr>
<td>Communicate Pro</td>
<td>0.12%</td>
</tr>
<tr>
<td>XMail</td>
<td>0.10%</td>
</tr>
<tr>
<td>Lotus Domino</td>
<td>0.07%</td>
</tr>
<tr>
<td>Microsoft Exchange</td>
<td>0.07%</td>
</tr>
<tr>
<td>NTMail</td>
<td>0.06%</td>
</tr>
<tr>
<td>DynFX</td>
<td>0.03%</td>
</tr>
<tr>
<td>Kerio MailServer</td>
<td>0.03%</td>
</tr>
<tr>
<td>GroupWise</td>
<td>0.02%</td>
</tr>
<tr>
<td>InterScan</td>
<td>0.02%</td>
</tr>
<tr>
<td>CMailServer</td>
<td>0.00%</td>
</tr>
<tr>
<td>Netscape Messaging Server</td>
<td>0.00%</td>
</tr>
<tr>
<td>Qmail *</td>
<td>0.00%</td>
</tr>
<tr>
<td>WinRoute Pro</td>
<td>0.00%</td>
</tr>
<tr>
<td>Unknown / Other</td>
<td>13.66%</td>
</tr>
</tbody>
</table>

* OSS software


Finally, Net Applications\(^{52}\) presents a data on market shares of web browsers and operating systems based on census-based surveys. The results presented by Net Applications come from a scan of the visitors browsers to the “exclusive on demand network of small to medium enterprise live stats customers”. According to the results of Net Application’s scan, MS Windows has a dominant position on the operating systems’ market (Figure A1.2) and MS Explorer is the most popular web browser (Figure A1.3).

Figure A1.2. Share of Operating Systems (data for October 2007)

Source: www.NetApplications.com

\(^{52}\) Available at http://www.netapplications.com
3. Country-focused studies

Concerning the geographical scope of available analyses, the most broad and advanced studies consider the European Union, the United States, Japan and Brazil.

European Union

The use of OSS in the European Union has advanced in part in relation to actions by public authorities. Certain local or regional European public administrations were among the early adopters of OSS and were already running OSS-based systems and applications in the late 1990s. Today, the European Commission services run a multitude of OSS-based applications. They initiate and sponsor new OSS projects for internal and external use. The rationale and benefits behind the strategy are to promote: broader choice of solutions; reduced captivity towards any particular product; faster and cheaper delivery of IT services by customisation and deployment of OSS solutions as opposed to full-scale development; and improvement of the IT staff skills through hands-on experience with emerging technologies.\(^{53}\) Open source related services are expected to reach 32% share of all IT services by 2010, and open source related activities might account for as much as 4% of European GDP by 2010 (European Commission, 2006).

Three major survey-based studies present recent OSS penetration rates for the European Union.

Méndez (2005) analyzed the adoption of OSS in European industry. He found that European firms have been recently increasing their adoption rates of OSS. Méndez estimated that at the end of 2005, 40% of EU firms were using open source solutions and another 8% had declared plans to introduce such systems. The telecommunications industry, mass-media and public sector entities were the most intense users of OSS, with 45% using OSS for key applications.

---

\(^{53}\) This information was based on a document obtained at the following location: www.ec.europa.eu/idabc/en/document/7389/5998 (last accessed in 2007). Current information is available from the Open Source Observatory and Repository for European public administrations at the following location: http://osor.eu/ (last accessed on 27.09.2008).
IDC (2005) presents the Western European Software End-User Survey of 625 firms. According to the results over 40% of respondents reported “significant, some or limited” use of OSS for operating systems purposes and nearly 60% declared use of open source databases.

The third study on the OSS penetration rate in the EU was done by MERIT (2006). According to the results presented, OSS market share is higher in Europe than in the US for some types of software (such as operating systems), which in turn is followed by Asia. These market shares have seen considerable growth between 2000 and 2005.

A survey run by MERIT in Germany, UK and Sweden differentiated responses by type of software and firm size. According to the results, the highest OSS penetration rates were found in Germany where 30.7% of small and 30.6% of large firms were found to use OSS operating systems. The smallest penetration rates were noted in UK large firms, where only 2% used OSS on desktops and 3.7% used OSS server operating systems (see Table A1.2).

<table>
<thead>
<tr>
<th></th>
<th>Germany</th>
<th>Sweden</th>
<th>UK</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>small</td>
<td>large</td>
<td>small</td>
<td>large</td>
</tr>
<tr>
<td>OSS as server operating system</td>
<td>30.7%</td>
<td>30.6%</td>
<td>9.8%</td>
<td>11.0%</td>
</tr>
<tr>
<td>OSS for databases</td>
<td>14.1%</td>
<td>20.8%</td>
<td>7.5%</td>
<td>8.2%</td>
</tr>
<tr>
<td>OSS on the desktop</td>
<td>13.7%</td>
<td>6.5%</td>
<td>3.4%</td>
<td>3.2%</td>
</tr>
<tr>
<td>OSS for websites</td>
<td>15.8%</td>
<td>17.3%</td>
<td>7.5%</td>
<td>8.7%</td>
</tr>
</tbody>
</table>


USA

Walli et al. (2005) present a study on OSS penetration rates in the US. They conclude that a clear majority of U.S. companies and government institutions are adopting OSS for some uses. Their survey results indicated that 87% of the 512 interviewed firms were using OSS programs. The usage of OSS goes beyond operating systems (i.e. Linux) to include various applications (Figure A1.4).

54 The threshold value between a small and a large firm in MERIT’s survey was 500 employees.
Walli et al. (2005) find that the bigger companies (larger than USD 50 million) are more likely to implement OSS solutions, with the telecommunications business being the leader in terms of OSS adoption intensity. The financial services industry reports the lowest OSS penetration rate. However, this is the industry with the highest number of declarations of planned introduction of open source software solutions.

**Japan**

There are no official data regarding OSS in Japan, although the OSS market is said to be expanding. The lack of data is mainly due to the difficulty of economic measurement for products that have a non-commercial character. However, reports from private firms provide some insights. For example, NEC Corporation, a strong promoter of Linux in Japan, estimates that the total size of the Linux-related business continues to expand by approximately 20% a year and will be 575 billion Yen in 2009 (Table A1.3).

**Table A1.3.** Estimated Linux relating business in Japan

<table>
<thead>
<tr>
<th></th>
<th>2005</th>
<th>2006</th>
<th>2007</th>
<th>2008</th>
<th>2009</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux System Integration</td>
<td>1,200</td>
<td>1,600</td>
<td>2,000</td>
<td>2,300</td>
<td>2,600</td>
</tr>
<tr>
<td>Support</td>
<td>900</td>
<td>1,000</td>
<td>1,100</td>
<td>1,300</td>
<td>1,650</td>
</tr>
<tr>
<td>Linux Platform (SW, HW)</td>
<td>700</td>
<td>900</td>
<td>1,100</td>
<td>1,300</td>
<td>1,500</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>2,700</td>
<td>3,500</td>
<td>4,200</td>
<td>4,900</td>
<td>5,750</td>
</tr>
</tbody>
</table>

*Source: NEC Corporation*

55 As of 4 October 2007, 100 million yen amounts to approximately USD 861.395 (http://www.oanda.com/convert/classic).
The most active adopter of OSS in Japan is the public sector, especially municipal governments. According to a survey on the introduction of OSS/Linux, the ratio for public sector entities with at least one OSS/Linux server is over 90% in 2005 (Table A1.4). Another survey covering local governments shows that municipals recognize the necessity of OSS literacy; over nine out of ten persons in charge of ICT at municipal offices want to participate in the OSS community.

<table>
<thead>
<tr>
<th>Table A1.4. OSS/Linux server introduction ratio, by sector</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Entities that have introduced at least 1 OSS/Linux server)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>---------------------------------</td>
</tr>
<tr>
<td>Public/ Public interest corporations</td>
</tr>
<tr>
<td>Financial/ Insurance industry</td>
</tr>
<tr>
<td>Service</td>
</tr>
<tr>
<td>Distribution</td>
</tr>
<tr>
<td>Assembly manufacturing</td>
</tr>
<tr>
<td>Processing manufacturing</td>
</tr>
<tr>
<td>Construction</td>
</tr>
<tr>
<td>Total</td>
</tr>
</tbody>
</table>


Contrary to municipalities, small and medium-size firms are sluggish in their OSS adoption. A survey of small and medium-size firms, of which annual turnover is between five hundred million and JPY 5 billion, counts no more than 6.7% as the share of Linux powered servers (OS) in 2007, as compared to a combined share of over 90% for Windows NT, Windows 2000 and Windows 2003 (Table A1.5). More than 60% of the firms state that they do not have the intention to adopt Linux as a server OS. This figure increased by 3.1 percentage points from the previous year’s survey. Over 50% of the firms declining to adopt Linux cite as a reason the limited number of engineers in their companies who have Linux literacy; around 40% say they are satisfied with Windows.
Table A1.5. Server operating system software, share of usage in small and medium size firms

<table>
<thead>
<tr>
<th></th>
<th>2006</th>
<th>2007</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(n=1876)</td>
<td>(n=2487)</td>
</tr>
<tr>
<td>Windows NT</td>
<td>18.8%</td>
<td>14.5%</td>
</tr>
<tr>
<td>Windows 2000</td>
<td>39.2%</td>
<td>38.5%</td>
</tr>
<tr>
<td>Windows 2003</td>
<td>25.9%</td>
<td>35.8%</td>
</tr>
<tr>
<td>Linux</td>
<td>5.5%</td>
<td>6.7%</td>
</tr>
<tr>
<td>UNIX</td>
<td>3.3%</td>
<td>1.7%</td>
</tr>
<tr>
<td>Netware</td>
<td>0.5%</td>
<td>0.6%</td>
</tr>
<tr>
<td>OS/2</td>
<td>0.5%</td>
<td>0.1%</td>
</tr>
<tr>
<td>Other</td>
<td>6.3%</td>
<td>2.0%</td>
</tr>
</tbody>
</table>


In the realm where software is used as a component in other products such as embedded software, OSS has begun to play an important role. All of the Japanese makers of non-CRT television sets, liquid-crystal display TVs and plasma display TVs have adopted Linux as a platform for embedded software. A similar transition is on the move with mobile phones. In January 2007, NEC Corporation, Panasonic Mobile Communications and NTT Docomo set up a foundation to promote Linux usage as a mobile phone platform, and now co-operate with Motorola, Samsung and Vodafone on this.

**Brazil**

Softex (2005) presents a comprehensive report on the role and impact of open source software in the Brazilian economy. The study outlines results of a few surveys on OSS penetration rates in selected fragments of the software market in Brazil. The report concludes that OSS is a growing phenomenon in Brazil, particularly within the Brazilian industry. The study points to the Linux operating system as the trigger for further adoption of other OSS solutions in a single firm.

Two other studies present statistics on OSS penetration rates in Brazil. The first one is the study by Fortes (2004), analysing 100 of the “most connected” firms according to *Info Exame* magazine. According to his results 64% of these companies already use Linux operating system. A newer study by Meirelles (2007) adds to the picture by showing some general statistics for the Brazilian economy as a whole. According to Meirelles’ results most of the Brazilian industry relies on non-OSS operating systems (Figure A1.5).

**Figure A1.5.** Operating systems, market penetration rates in Brazilian firms (2006 and 2007)

A similar situation to the market for operating systems was found by Meirelles (2007) for spreadsheets used by Brazilian firms, where a large majority uses a proprietary program (Table A1.6).

**Table A1.6.** Spreadsheets used by Brazilian firms (2006 and 2007)

<table>
<thead>
<tr>
<th>Spreadsheets</th>
<th>Penetration rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Excel</td>
<td>92%</td>
</tr>
<tr>
<td>Open and Star Office</td>
<td>6%</td>
</tr>
<tr>
<td>Other</td>
<td>2%</td>
</tr>
</tbody>
</table>

*Source: Meirelles, 2007.*
## OECD PROJECT ON INNOVATION IN THE SOFTWARE SECTOR

### Glossary (for draft report chapters)

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application program interface</td>
</tr>
<tr>
<td>ASP</td>
<td>Application service provider</td>
</tr>
<tr>
<td>B2B</td>
<td>Business to business</td>
</tr>
<tr>
<td>B2C</td>
<td>Business to consumer</td>
</tr>
<tr>
<td>BAM</td>
<td>Business Activity Monitoring</td>
</tr>
<tr>
<td>CRM</td>
<td>Customer relationship management</td>
</tr>
<tr>
<td>EAI</td>
<td>Enterprise Application Integration</td>
</tr>
<tr>
<td>ERP</td>
<td>Enterprise Resource Planning</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>HCM</td>
<td>Human Capital Management</td>
</tr>
<tr>
<td>ICT</td>
<td>Information and Communication Technologies</td>
</tr>
<tr>
<td>ISO</td>
<td>International Organisation for Standardization</td>
</tr>
<tr>
<td>ISV</td>
<td>Independent software vendor</td>
</tr>
<tr>
<td>MNEs</td>
<td>Multinational Enterprises</td>
</tr>
<tr>
<td>MSP</td>
<td>Managed Service Provision</td>
</tr>
<tr>
<td>OSS</td>
<td>Open Source Software</td>
</tr>
<tr>
<td>PDA</td>
<td>Personal Digital Assistance</td>
</tr>
<tr>
<td>SaaS</td>
<td>Software as a service</td>
</tr>
<tr>
<td>SOA</td>
<td>Service-oriented architecture</td>
</tr>
<tr>
<td>UCC</td>
<td>User-Created Content</td>
</tr>
<tr>
<td>VPN</td>
<td>Virtual private network</td>
</tr>
<tr>
<td>VoIP</td>
<td>Voice over Internet Protocol</td>
</tr>
<tr>
<td>WLAN</td>
<td>Wireless Local Area Network</td>
</tr>
<tr>
<td>WiMAX</td>
<td>Worldwide Interoperability for Microwave broadband</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
</tbody>
</table>
REFERENCES


Dependability Computing Programme, Computer Science Department, University of Maryland, College Park, Maryland, USA.


Fellers, J.W. and R.P. Bostrom (1993), Application of group support systems to promote creativity in information systems organizations, System Sciences, Proceeding of the Twenty-Sixth Hawaii International Conference.


MERIT (2006), Economic impact of FLOSS on innovation and competitiveness of the EU ICT sector, UNU-MERIT, Maastricht University.


Softex (2005), Impact of the Free Software and Open Source on the Software Industry in Brazil, Available at: http://www.softex.br


Techconsult (2006), "Der deutsche Markt für Linux und Open Source (German Market for Linux and Open Source)" TechConsult GmbH, Available at: http://www.techconsult.de


