J Internet Serv Appl (2011) 2:11–22
DOI 10.1007/s13174-011-0019-x

ORIGINAL PAPER

How the Internet transformed the software industry

Anthony I. Wasserman

Received: 7 February 2011 / Accepted: 14 April 2011 / Published online: 11 May 2011

Abstract The growth of the Internet has had a huge impact on the software industry, from the ease of creating new businesses to the processes that companies use to develop, distribute, and support their products. Software architectures and platforms for web applications differ significantly from traditional desktop and client-server applications, and require a new generation of programming languages and development tools. Developers make extensive use of open source software, particularly at the infrastructure level of their applications. Development teams can easily use Internet-based tools for collaboration, thus facilitating distributed organizations. Product marketing now emphasizes attracting prospective customers to the company's website. Companies can release products over the Internet, or provide a hosted web application, both of which are more efficient and less expensive ways to sell their products. Companies can also support their products over the Internet through online discussion forums, often with users helping one another. Ongoing advances in mobile and cloud computing, styles of user interaction, and software business models are certain to have a large effect, leading to innovative new products from both new and established companies.

Keywords Internet · Software industry · Software development · Software products · Business models · Open source · FOSS · Web applications · Hosted applications · Cloud computing · Mobile computing

A.I. Wasserman (✉)
Carnegie Mellon University Silicon Valley, Moffett Field, CA, USA
e-mail: tonyw@sv.cmu.edu

1 Introduction

The impact of the Internet on the world is incalculable. It has revolutionized communications among organizations and individuals, made it possible for millions of people to purchase almost everything they need online, enabled the creation and expansion of hundreds of thousands of businesses, and made information available to billions of people in ways that were previously impossible.

Along the way, the Internet and resulting web sites and applications have disrupted many existing business practices, and many industries. Books and magazines, previously available only in "hard copy" format, are quickly being replaced by electronic formats, along with specialized devices for viewing them. Media, such as music, television programs, and movies, are similarly moving to digital format, with a marked decrease in the sale of CDs and DVDs. For example, according to compete.com, a leading web traffic analysis site, the collaboratively-edited Wikipedia website currently attracts more than 80 million unique visitors a month, while the use of traditionally-edited print encyclopedias dwindles. Consumers have immediate access to comparative price information from competing merchants, and can use that information in their shopping decisions.

Finally, people from all over the world can make personal connections and establish groups of shared interests on virtually any topic. People can meet one another "virtually" for social reasons, to share knowledge about a topic (e.g., travel or health), or to organize in support or opposition to a cause. It is quite common for people to find one another after many years of not being in contact.

Of course, there are both positive and negative aspects to this situation. On the positive side, people can easily establish and maintain contact with friends, family, and colleagues in ways that simply weren't possible twenty years

ago. On the other hand, the massive amounts of personal data stored on websites makes it possible for aggregators to assemble extensive dossiers on individuals, and make it available to others, giving those individuals little or no opportunity to edit the data or block access to it.

In summary, the Internet has affected every person and industry in advanced economies, as well as many of them in less-developed economies, even those without direct access to the Internet. Companies have adjusted their business methods and models to accommodate the Internet. Thus, an automobile dealership will post listings of their available vehicles and will recognize that many of the potential customers entering their stores will have already done extensive research on the vehicles and on their prices.

The software industry has been strongly affected by the ongoing development of the Internet. The goal of this paper is to illustrate not only the "classic" structure of software businesses, but also the forces that have radically transformed the industry since the advent of the Internet, leading to a discussion of various "modern" software business structures. Section 2 addresses the impacts on various departments typically found in a software company: R&D (engineering), quality assurance, marketing, sales, and customer support. Section 3 addresses the growth in open source software and the way that it has affected software products, development practices, pricing, and more. Section 4, in conclusion, projects the ongoing impact of these factors on the software industry, with an eye toward its future.

2 Software and software companies

Software was developed and distributed before there was a software industry. In the US, much of the software that was developed by or for a government agency was placed into the public domain for anyone to pick up and use. Much of that software was of little value, since it was often written in assembly language or obscure programming languages for highly specialized machines. The effort to adapt it to a different environment might well be greater than the cost of rewriting the software from scratch.

Software was also frequently exchanged through user groups sponsored by computer vendors, of which the best known were IBM's SHARE and Digital Equipment Corporation's DECUS. People would contribute software to libraries managed by these organizations, making them available to anyone interested in using them. It was not uncommon for people to fix or enhance such software, and contribute the new version back to the user group.

The commercial software industry emerged soon after IBM's 1964 introduction of the System/360 family of computers, with the OS/360 operating system. IBM consolidated their business-oriented computers and their scientific computers into a single product line. More significantly from a software perspective, it was now feasible for anyone to write an application that would run across the entire product line, from the smallest System/360 machine to the largest. The System/360 family quickly dominated the market, and thus created a sizeable installed base of machines.

From a commercial perspective, the most successful of the early third-party software applications were for data management. IBM took note of this development, and unbundled their pricing, charging separately for some of their applications, notably the IMS data management system and their programming language compilers. These early commercial successes led to today's huge software industry.

2.1 Software companies and products before the Internet

Prior to the widespread emergence of the Internet, there were three major types of software businesses: enterprise, consumer, and shareware. While some vendors of enterprise software also sold consumer-oriented products, they followed different processes for products in the different categories. There was also some free software, such as the GNU compilers and the X-Window System, available from governments, nonprofit organizations, and research groups.

Enterprise software products were licensed to organizations. Because the price for such products usually was thousands of US dollars (or equivalent), there was often a lengthy sales process that required a dedicated sales force to call upon the prospective customer in person while they evaluated the product (and one or more competitors). Consumer software products were sold to individuals, and were prepackaged for sale through retail channels or directly from the vendor. The price was below one thousand dollars, and usually below a hundred dollars, with consumers learning about the product through published reviews, articles, and vendor marketing programs. Shareware was also aimed at individual buyers, and typically developed by individuals or small teams, and was sold for very low prices directly from the developer. Some of these shareware products, such as WinZip, attracted many customers through word-of-mouth and favorable reviews in computing publications, to the extent that their developers were able to build an ongoing business. Customers of enterprise software typically also paid an annual maintenance fee of 15–20% of the license price, which entitled them to customer support and product updates; consumer software typically offered fee-per-call support for products and a discount on new versions.

In all of these cases prior to the development of the Internet, though, the software vendor had to deliver physical media to the customer: tape, floppy disks, or compact disks. Software companies had to design packaging, manufacture media, and ship the software to customers, retailers, and/or distributors. Those steps, as we shall see, had a significant impact on software development.

Consumer software products, including shareware, ran on local machines without making connections to external machines or networks. While some of these products were component-based and used proprietary protocols, such as Microsoft's OLE, all of the resources needed for running the program were local to that machine. Enterprise software products made more use of distributed computing architectures. These architectures include IBM's System Network Architecture (SNA), Microsoft's Component Object Models (COM and DCOM) and the Object Management Group's Common Object Request Broker Architecture (CORBA). These architectures were often at the heart of client-server applications, where client workstations accessed functionality and data stored on another machine (server). Once again, though, these applications ran within a customer's network.

2.2 Software companies and products in the Internet age

Today's software companies and products are quite different from those of the 1980s and 1990s. The difference is not so much in the types of companies, since there are still enterprise, consumer, and shareware businesses, but rather in the way that they market, sell, and deliver their products, and the ways by which they interact with customers and prospects. The Internet was (and still is) the driving force behind these changes, since it enables individuals to interact with the company and its products on the Web and enables electronic delivery of software products and updates.

The widespread availability of the Internet has led to even bigger changes in software products themselves. First, there are some new classes of applications. The first of these are hosted (or cloud-based) applications, originally termed Software as a Service (SaaS), which does not require any customer installation of software. Hosted applications are available for a wide range of domains, including email, personal productivity, office automation, customer relationship management, software development, online communities, e-commerce, telephony and conferencing, and massive multiplayer games. Hosted applications are usually sold by monthly or annual subscription, though many are free to the user and funded by third-party advertising and/or by the products and services sold through the application.

The second new class of application, an "app," is available for modern mobile devices and sold through app stores that aggregate apps from many different sources. Many of these apps are free, with only a tiny percentage currently selling for more than $10 US. Apps other than games are rarely from software companies, but predominantly from sellers of goods and services. These sellers lack the internal organizations for selling software, which makes the app store concept valuable. While Apple's AppStore led the way, other vendors of mobile devices and platforms have followed their lead, including Nokia, RIM, Microsoft, and the Android Market. Apple's recent introduction of an AppStore for MacOS applications, combined with their announced plan to remove application software from their retail stores, may be viewed as an indication of their intent to deliver all of their software over the Internet.

The third new class of application is free and open source software (FOSS),[1] much of which is noncommercial. There are also several hundred commercial FOSS developers and vendors that make money by selling services, e.g., training and updates, for the FOSS software. As with hosted applications, FOSS software is available for a broad range of applications, with particularly strong choices for software development, as well as for system infrastructure and management. FOSS offerings are increasingly competitive with traditional enterprise and consumer applications.

None of these types of software products would be possible without the Internet. In the first two cases, the running applications rely on the Internet to perform their functions. In the FOSS case, the Internet removes many of the costs associated with software sales and distribution, as well as facilitating collaboration among members of the development team, who may be geographically separated.

Enterprise, consumer, and shareware software products also make extensive use of the Internet. From the perspective of the software vendor, three uses of the Internet are particularly important. The first of these is registration and ongoing online validation of user licenses; such validation can sharply reduce unauthorized use of software products, though it should be noted that there are also numerous online services that provide pirated software and license key generation mechanisms. The second of these is digital delivery of products and their updates. Most low-cost software products are available only by download. Application vendors routinely include "call home" features in their applications or in related applications automatically called from the primary application. These features are used not only for license monitoring, but also to offer and/or deliver updates to the application.

Games are an important category for consumer-oriented online software products. For example, millions of people play multiplayer online games from companies such as Blizzard Entertainment (World of Warcraft), Zynga (Farmville, Cityville), and id Software (Doom, Quake). Many games are hybrid, running locally on either a personal computer or dedicated game console, as well as over the Internet. Doom is widely recognized for having proven the validity of the shareware business model [15]. Game developers have been highly innovative with their products, being among the first to enable eCommerce during play and the first to offer products for mobile devices. Games consistently dominate the

[1] Some people prefer the equivalent term "FLOSS" (Free, Libre, and Open Source Software).

lists of best-selling apps in the application stores for mobile platforms.

The third major use of the Internet by software vendors is to reduce the costs of providing product support. While enterprise software customers can still pay for telephone support, as well as premium levels of support, e.g., access to specialists or guaranteed response times, most users are directed to "self-service support," typically consisting of online access to frequently asked questions (FAQs) and to online forums on various product-related topics, e.g., installation, platform compatibility, and problem workarounds. Vendor support representatives monitor and respond to issues raised in such forums, not only to assist customers and to identify important fixes and new features, but also to maintain a level of civil discourse and to protect the vendor's reputation.

As with open source communities, commercial vendors have created communities of their users around their applications in the hope that users will help one another in solving their problems. Many of the larger software companies have user groups, which attract thousands of professionals to annual meetings to meet with company executives and technical leaders. As another example, salesforce.com has developed the force.com platform, making it feasible for independent developers to create applications that work with the salesforce.com application. These applications are then available (free or for a price) to other salesforce.com users.

In summary, modern software companies make extensive use of the Internet in every aspect of their businesses. The Internet influences the products that they build, as well the ways that they develop, sell, market and support them. Companies that failed to adapt to these transformative changes to the software industry did not survive the transition, as new companies have arisen to displace them and to create innovative applications. (Development of the Internet has also had a similar impact on embedded systems in telematics, consumer electronics, and telecommunications, but that topic is beyond the scope of this paper.) We next review the ways that different departments within a software company have been affected by the Internet, and briefly discuss how these changes have modified existing business models and led to the creation of new ones.

2.3 Software development in the Internet age

New types of application and technological advances lead to new tools for application development. For example, the Unix operating system was built with the C programming language, making C the de facto programming language choice for Unix applications. Because C included some low-level programming constructs, it was suitable not only for traditional systems applications, but also for embedded systems applications, where it remains a popular alternative more than 35 years later. Many software development tools [14, 22, 29] were built to run on Unix systems. The Smalltalk-80 environment [8], drawing upon the earlier Interlisp environment, was a strong influence on future programming environments.

Numerous development environments were built for the Windows platform. Borland created JBuilder and C++ Builder for Java and C++ programming. Similarly, there were many successful development tools, including PowerBuilder, Progress, and Uniface, for building client-server applications. Such tools, termed 4GLs (Fourth Generation Languages) simplified the creation of applications connecting a database on a server machine to a graphical user interface on the client (desktop) computer.

Over time, Microsoft's Visual Studio programming environment became the dominant programming environment for Windows applications, with Apple Xcode as the primary environment for MacOS applications. These environments built successfully on lessons learned from several previous generations of programming environments, providing syntax-directed editing, formatting of code, and powerful tools for tracing and debugging. Many of the design principles found in these editors were subsequently incorporated into development environments for HTML, PHP, and other languages used in the development of web applications.

Developing such web-based applications involves design of the "front end," or client side, with the user interface, as well as the "back end," or server side, with program logic and data storage. The skills and technologies needed for these different aspects of development are quite different. Many simple web sites, i.e., those with no server side components, can be developed with HTML and JavaScript, either by hand coding or by using site-building tools such as Adobe Dreamweaver or RapidWeaver.

Server side development can range from simple Common Gateway Interfaces (CGI) for processing a web form [23] to complex backend content management systems, massive multiplayer online games, social networking sites, or highly scalable, distributed e-commerce applications that perform all of the processing and data management as well as dynamically creating the HTML output seen by the site users. Many types of applications, ranging from personal income tax management and image editing to sales force automation and electronic medical record systems, that were formerly developed as standalone applications have been implemented as web applications.

With the increased experience in developing these various complex web applications has come a broad range of languages and frameworks to aid with future development. For example, there are more than a dozen commercial frameworks ("engines") for game development [5]. Sun Microsystems developed the Java Enterprise Edition framework for building high-volume scalable web applications; Johnson developed the open source Spring framework [11] as an alternative.

Among the programming languages developed for development of server-side functions are Perl [25], PHP [16], Python [17, 27], Lua [10], and Ruby [6]. While C and C++ are compiled to object code on the host computer, and Java is compiled to an interpreted byte code, these newer languages are interpreted at run time. PHP has been designed to work effectively as part of an infrastructure with the Apache HTTP server [1] and a relational DBMS such as MySQL. Ruby was specifically designed for the development of data-driven web applications, and is frequently used with the Rails framework that provides a straightforward way to support the development process [24]. Specialized development environments, such as the Zend framework for PHP [21], help to reduce the amount of custom code to be written for an application.

The recent emergence of applications for mobile devices, including mobile web applications, has led to additional tools and environments. Microsoft has introduced Expression Studio and expanded Visual Studio to assist Windows Phone app developers. Google has introduced a software development kit (SDK) to support Android development using Eclipse. Rhomobile has introduced Rhodes, a Ruby-based environment for cross-platform mobile development.

The above list of examples is far from complete, and is simply intended to be indicative of the extensive advances in development environments and programming languages for Internet-based applications. These tools, along with the distributed architecture of web-based applications, have strongly affected development processes and organizations, as well as project governance [3].

2.3.1 The structure of development organizations

The dominant model for software development has long been a colocated team of people responsible for the concept, design, development, testing, and ongoing maintenance of software products. That model has applied not only to software development for commercial licensing, but also to internal corporate applications. It is very easy to visualize the typical software development team and their shared workspace. Each developer's computer was equipped with licensed commercial software, enabling them to allocate development tasks, to write and test the code, to maintain versions of the software, and to share their work. The development organization would adopt a process and schedule for its work, and regularly report progress to management.

The Internet has disrupted this model in several important ways. More important, the Internet has vastly improved global communication, making it feasible to have remote developers or development teams. Since the mid-1990s, companies have engaged in outsourcing, hiring developers in locations far removed from the company's home base, to do the needed software development. While much of this work has been done in India, it is by no means the only location for outsourced software development. Another approach has been to separate a company's development team from the company's sales, marketing, and administrative functions. In the former case, there is a contractual agreement between the software vendor and the development organization, while, in the latter case, everyone works for the same company. In each situation, though, the Internet enables the company to reduce its costs for software development by hiring developers in a lower-cost location.

There are numerous variants on these approaches, based on the observation that there are smart and talented people all over the world. Many large companies have created development teams in different geographical locations, giving each team the responsibility for specific tasks or products. Such decisions are made not only for economic, but also for political and social reasons. While there may be cultural and communication issues associated with such highly distributed teams, there is now substantial experience in addressing such problems. Along the same lines, it is often the case that a company wants to hire a particular individual, but that the individual is unable or unwilling to move to the company's primary development location. Again, the Internet enables such an arrangement, though the individual in question may have to adjust his life to accommodate the company's primary time zone.

This author once managed a distributed product development group based on the West Coast of the US while reporting to a manager and executive team on the East Coast of the US. The development team, with members on both Coasts and one in the UK, shared a project repository with code and documentation. Team members, who had no desire to relocate across the country, communicated by Internet Relay Chat, instant messaging, telephone conference calls, and email, with occasional in-person meetings. (Today the team would be more likely to use Skype VoiP and video conferencing, which were not available at that time.) Despite the communication and travel overhead, this team easily adjusted to the Internet-dependent process and delivered high-quality open source software (meeting time and budget expectations) that enabled developers to create mobile web applications for a Java Enterprise Edition application server. (Interestingly enough, the first customer was a UK-based system integrator building an application for a European company.) Such an approach is now the norm for many established software development organizations, taking advantage of Internet-based resources to enable distributed development.

2.3.2 Software release schedules

Another important difference in software development in the Internet era involves the frequency of product releases. For

many years, many software companies aimed to make one major and one minor release each year, where the minor releases would address bugs and platform compatibility issues and the major releases would also add new functionality to the product. The costs of manufacturing the product media and shipping the new versions to customers or the retail distributors meant that additional releases meant additional expenses. While companies would sometimes create intermediate releases for selected customers to address a serious product issue, that was an unusual step, and most customers had to wait for the normal release cycle for such product fixes and enhancements. To prevent such problems, software companies made significant investments in testing and quality assurance.

The advent of the Internet made it feasible to distribute software electronically, thus greatly reducing the cost of distributing new versions of software. Fittingly, Netscape, developer of one of the first browsers, was the first software company to recognize this opportunity, releasing 10 versions of Netscape Communicator 4.0 through 4.08 in an 18-month period beginning in June, 1997. Users could download and install these successive versions from Netscape's download sites. Netscape's development organization recognized that the Internet made it possible to create releases much more often than had previously been feasible. Product bugs become less critical, since it became relatively easy to create a new version of the product and make it available for download. Thus, version 4.01 of Netscape Communicator was released only a week after version 4.0, a schedule that would have been completely impractical before the Internet. The next three releases followed at one-month intervals [18].

The software development and release process has now changed for products that can easily be updated in this manner. Even software products in their early development stages are often released as early as possible, with frequent updates as needed. For example, the Skim PDF Reader for Mac OS X had 18 released versions during 2007, 16 of them prior to its official 1.0 release, and has had 53 releases in all as of January, 2011 [26]. Along with the Linux kernel, Skim is among the best examples of the agile development tenet of "release early, release often."

In addition, hosted applications can be updated as frequently as desired by the software vendor. Since popular cloud-based applications are hosted on a multitude of machines, it is easy to test a new version of the application on a small subset of the user population and to do a "rolling" release of the new version, or to roll it back to the previous version in the event of an unforeseen problem. At the height of its growth about 10 years ago, the eBay Internet auction site planned as many as 20 incrementally enhanced versions of their site each year.

In short, it is clear that the ability to make new versions of software available quickly and cheaply over the Internet has significantly changed software development practices. With respect to enterprise software applications, though, it should be noted that IT organizations are much slower to update installed software and often prefer to defer such updates until it becomes absolutely necessary to do so. As one notable example, many organizations are still using Microsoft Windows XP, even though there have been two subsequent major releases of the Windows operating system; in recognition of this situation, Microsoft repeatedly has extended their termination date for support of Windows XP, now scheduled for 2014.

2.3.3 International markets

The Internet has greatly enabled the ability of people to find products outside their local markets. While that phenomenon has affected books and music more strongly, the software industry has also been significantly influenced.

Internet search mechanisms enable people to find, download, and purchase software products wherever they may be in the world. Internationally recognized payment mechanisms, including PayPal and major credit cards, are widely accepted for software purchases, so the remainder of the process is often just a matter of transferring bytes over the Internet. While issues of customs duties and economic sanctions occasionally serve to limit such transactions, the commercial opportunities for vendors of low-cost, consumer-oriented software can be quite significant. That statement is especially true for software vendors based in smaller and less developed countries, where the Internet opens up much larger markets. For example, the OrangeHRM application for human resource management and the WSO2 web services platform were both developed in Sri Lanka.

Many software products are now built with these international opportunities in mind. Microsoft's Windows 7 operating system, for example, is available with 35 different language packs. Similarly, Adobe directly supports more than 20 languages for its Photoshop product, and works with an internationalization (i18n) partner to support other, less commonly used, languages. As a result, many software products are developed with i18n and localization (l10n) support, making it possible for menus, online help, and other messages to be translated into multiple languages.

2.4 Software product marketing in the Internet age

Software product marketing involves activities that include generating awareness of a company's product(s), describing the product features in a manner that will attract people who become aware of the product, presenting its benefits in comparison to current practices, and enumerating advantages of the product when measured against its competitors. The sales process is more specifically focused on convinc-

ing the prospective customer to spend money on the company's products and ancillary services. The sales process for a complex enterprise software product may include presentations, demonstrations, and product evaluations, as well as price and contract negotiations, often involving many people from the software vendor meeting with many different people in the customer's organization.

Before the Internet, all of a company's product marketing was *outbound,* and included such lead generation activities as direct mail, advertising, press releases, product launches, trade show booths, and meetings with industry analysts, just to name the most common approaches. These activities were expensive, took a long time to prepare and execute, and often provided the vendor with very little data as to which specific events or publications provided the greatest return for the money. It was not unusual for smaller software companies to spend more than half of their revenues on sales and marketing programs.

The Internet has opened up a huge range of new possibilities for interacting with prospective customers. Of course, the company's web site is central to the modern marketing program. The site displays basic information on the company and its products, conveys the brand image, provides the links to product support and forums, and may include the ability to view product-related videos, download white papers and/or a trial version of products, and possibly even purchase product licenses directly from the company.

Given the central role of the company website, a key goal of any marketing campaign is to drive likely prospective customers to the website. There are many different approaches for achieving this goal through Internet-based *inbound* marketing [9], including ad placement on other sites, sponsorship of email newsletters, use of webinars, social media, online video, and, especially, search engine optimization, including the purchase of keywords that will display an ad for the company in response to user searches. Companies selling a certain type of software will sometimes not only purchase terms related to their product category, but also the phrases and names associated with competing products. All of these links can be tagged so that the logs for visits to the company's site include information about the referring site. In that way, it is possible to quickly identify which of the marketing programs and purchases are most effective in bringing traffic to the company's site [12].

Once the prospective customer is viewing pages on the company's web site, it is very common practice to track the user's behavior. At a minimum, the company can set cookies on the visitor's machine that will retain information about that visitor through future visits to the site. Someone who visits a company's site repeatedly has a higher probability of being a candidate customer than is someone who visits only once. There are now sophisticated marketing automation systems (from Eloqua, Marketo, Pardot, and other vendors) to score the user's movements on a site. Points can be assigned to the visitor for such activities as registering for a webinar, downloading a white paper, joining a mailing list, or for providing the name of one's employer. Once the point total reaches a specific level, visitors will receive a telephone call or email message from the company to qualify them further as a prospective customer.

These marketing techniques would not be possible without the Internet. While they can be expensive in absolute terms, these approaches are much more sharply focused than are traditional marketing programs, and reduce the cost to find each qualified prospective customer. Furthermore, associated analytic tools make it possible to quickly evaluate the value of each marketing tactic, and to adjust the program accordingly before spending large sums of money on ineffective programs. Similarly, it is quite easy to compare the results among several different marketing messages or ads, cancelling the less effective ones.

Finally, international opportunities created by the Internet also affect product marketing programs, since the techniques used to generate awareness of the company and its products are often seen globally. Many companies, particularly those whose local language is not English, purchase domain names for their company in various national top-level domains, create multilingual web sites, and purchase search terms in multiple languages.

The result of Internet-based marketing has been a marked shift in the way that software companies spend their marketing budgets, allocating higher and higher percentages of these budgets to online programs. As a result, many of the software-focused trade shows and publications have downsized, disappeared, and/or moved to the Web.

2.5 Software product sales in the Internet age

Sales programs pick up where the online marketing leaves off. Since modern marketing programs do a better job of identifying the most likely prospective customers, salespeople can now spend their time more effectively. While the time and cost needed to complete a major sale of enterprise software is still high, the sales team can focus on the best prospects and use the Web or a less-expensive telephone sales team for smaller opportunities. If a prospective customer can download product documentation with a 30-day trial of the product, and can obtain online support for that product, then it is often possible to defer engaging the sales team until the prospective customer is convinced about the merit of the product and is ready to pay for the product. Thus, the online services also serve to shorten the sales cycle, i.e., the period of time between initial engagement and purchase.

As a general rule, enterprise software sales still need a sales team to present the features and functions of complex software applications, to address competitive issues, and to

negotiate pricing and licenses. The corporate prospective customers for such products want to work with the vendor's representatives to work through these issues and more.

However, that is rarely the case for individuals licensing consumer-oriented software. The sharp reduction in the number of retail outlets for software products has driven vendors to enhance their direct sales channels. This disintermediation process has been valuable to software vendors, since it allows them to retain a higher percentage of the selling price, receive their payment more quickly and to establish a direct connection with the customer.

While customers can still purchase packaged software products through online retailers, they can also download trial versions directly from the vendor, with the option of converting that trial into a licensed copy through an online payment mechanism. In general, prospective customers can contact vendor sales representatives by telephone or email for pre-sales information, and can also find product information on the vendor's web site. Some vendors have established e-commerce capabilities in many different countries, and can thereby reduce their dependence on international distributors.

As noted in Sect. 2.2, many vendors of consumer-oriented mobile applications have completely eliminated their sales organizations, particularly when an "app store" is the only available sales channel, as is the case with iOS apps. In some cases, mobile app vendors can also use third parties as resellers of their mobile apps; Handango and Amazon.com have established such businesses.

Beyond these differences, many packaged software products are being replaced by hosted applications, as noted above. For example, the two leading US vendors of consumer income tax software offer both a packaged product and an online service for tax preparation. Beyond that, however, is the beginning of a broader transition away from packaged applications toward hosted services. For example, Intuit acquired mint.com as a hosted alternative to its packaged Quicken application.

Hosted applications also offer vendors the possibility to sell variants of the software more easily than with the packaged solution. For example, the BasecampHQ project management tool is offered in five different plans, from a free plan suitable for an individual to four paid levels supporting larger numbers of projects and users.

The combination of free and paid hosted services, commonly termed "freemium," has become quite common as a business model for consumer-oriented software. For example, Dropbox allows anyone to store and share 2 Gb of data on line for free, and also offers paid premium services for those who want to manage up to 100 Gb. (Such a product could not exist without the Internet.) Other online services, such as Skype and the Flickr photo-sharing site, also have free and paid versions of their services.

Another Internet-enabled business model involves ad-supported hosted applications. One approach gives the user the option of using a free version of the application, in which ads are presented on the screen, or a paid, ad-free version. Ad-supported applications are increasingly common for mobile apps, and help developers to generate revenue above and beyond the typically low sales price of the app itself.

Consumer software sales have also been strongly affected by social networking resources. Before the Internet, users had very little information about the functionality, quality, and performance of packaged software, apart from reviews published in consumer-oriented computer magazines, which rarely reflected typical consumer situations, and were sometimes compromised by potential conflicts with advertisers. Today, by contrast, product reviews can be widely found on the Internet, not just for computer software, but for a wide variety of consumer-oriented goods and services. Many vendors of both packaged and hosted applications include consumer reviews on their own websites. Both Apple's App-Store and the Android Marketplace include individual reviews and overall summaries along with each app, free or paid, in their stores.

2.6 Software product support in the Internet age

Software support covers a broad range of activities, with the two most common ones being to assist users with product installation and to coordinate reports of product faults of various types. As more software applications are hosted by the vendor rather than installed on customer sites, the need for installation assistance (and its associated costs) will diminish, shifting the primary product support focus to addressing product quality issues.

The support organization is often the first to hear about quality issues in released software products. Before the Internet, they could track these problem reports, and work with the product development team to assess their importance, helping to determine which issues should be addressed in each major and minor release of the software product. In the case where a severe problem made it impossible for a customer to use the product, the support team would work with the development organization to produce a patch or a product update outside the regular product release schedule, and ship that fix to the affected user(s).

Today, it is much easier to create and deliver program updates and fixes, often on short notice, and often to address relatively minor problems. Operating system vendors have been particularly diligent about updating their products. Microsoft, for example, regularly schedules a set of updates for release on a Tuesday. Many users of Microsoft's products subscribe to their automatic program update services, and thus automatically receive such updates.

In general, software product companies assume that their customers will be running a recent, ideally the current, version of their product. Customers who contact the company for product support are often strongly encouraged to update their installation to the latest version of the product.

Another key difference in product support comes from the widespread availability of product support forums to assist customers with their questions and problems. Such a forum is intended to augment online help and product documentation, and to reduce the number of support calls and email messages received by the product vendor's support staff. In general, these product forums follow a "users helping users" approach, with the vendor's support staff monitoring them to address the most serious unresolved issues and to eliminate highly negative postings. In addition to the vendor's own support forums, there are often third-party support sites, such as techsupportforum.com for the most widely used software products.

In summary, the Internet has provided significant benefits to software product companies, using online support forums to reduce the cost of providing product support and to create a community of interest around their product(s) in a way that encourages users to assist one another.

3 The impact of open source software on the software industry

Until now, this paper has focused on the direct impact of the Internet on the software industry, but the discussion would not be complete without noting that the Internet has had a major impact on the development of free and open source software, which has, in turn, been very influential in transforming the software industry.

There has long been free software, dating back to the earliest days of the computer industry. Universities and research laboratories, both commercial and government-based, frequently provided their work at no charge to interested parties. Many of these programs were written in machine-oriented languages to run on special purpose hardware. Each computer vendor used different and incompatible architectures and instruction sets, so a program written for one of these machines would not work on other machines. This incompatibility problem was reduced after the release of FORTRAN and COBOL in 1957 and 1960, respectively, but many programs written in these languages still had machine dependencies because of differing computer architectures. As noted above, there was no viable market for software products until the mid-1960s.

Even after the software market emerged, many groups continued to give away software. For example, the BASIC language system was developed at Dartmouth College, which gave out the BASIC compiler to schools and universities as a teaching tool [13]. Artificial intelligence laboratories at Stanford and MIT, among others, shared software developed in LISP. Richard Stallman created the GNU project at MIT [7], and distributed that software to anyone who wanted it, in keeping with his beliefs in making software free. He subsequently created the Free Software Foundation (FSF), which emerged as a leading advocate for free software. The Open Source Initiative (OSI) [19], created in 1998, produced the Open Source Definition and approves software licenses that are compliant with that Definition, including those created by the FSF.

Thousands of high quality software products and components have been developed and released under the licenses approved by the FSF and the OSI, including many that support the open standards of the Internet itself. This constantly growing body of high-quality software, available to anyone at no cost, has provided a strong contrast to the traditional proprietary software. Some open source products, such as the Apache HTTP server and the Eclipse development environment, are the dominant products in their field, while others, including the Linux operating system, the OpenOffice.org productivity suite, the MySQL database management systems, the JBoss middleware, and the Firefox browser, hold significant market share in segments long dominated by proprietary software.

This rapid growth in free and open source software has also transformed the software industry, with the Internet playing a major role in facilitating this growth. First, the Internet enables communication and collaboration for everyone, supporting community-based, noncommercial software as well as proprietary software development. Anyone can start a FOSS development project and can create and host a project repository for free on any of numerous "forges." Anyone wanting to join a project can easily search the major forges for a suitable project and begin to contribute. Anyone wishing to use such software can download project code and try it out.

The major forges (e.g., SourceForge, Google Code, and GitHub) host hundreds of thousands of open source projects of varying maturity and quality. In many cases, users are able to obtain paid product training and ongoing support for these projects from their developers or from third-party service providers. In addition, the Mozilla, Eclipse, and Apache Foundations host several well-established, widely used open source projects that compare favorably to their commercial competition. These projects, enabled by the Internet, are able to attract and engage contributors and other community members from around the world.

In addition, single-vendor open source companies offer open source end user applications, system management tools, and infrastructure software for which they provide a traditional range of paid support services. Community versions of their software are available at no charge (often in

both source and binary installer formats), as are their online documentation and forums. Some companies have only a single version of their software, while others follow an "open core" model, providing a community release of the core version, and offering proprietary premium features using a commercial license.

As a result, proprietary software companies are no longer only in competition with one another, but also with community-based FOSS projects and single-vendor commercial open source products. Since people can use FOSS forever at no cost, it can be hard for some organizations to justify licensing and paying for proprietary software with similar functionality. Some governments have developed explicit policies either favoring the use of open source software or, at a minimum, making sure that FOSS options are considered in software procurements.

During the recent economic downturn that affected many of the wealthier countries, many corporations gave serious consideration to FOSS for the first time, largely as a means to reduce their software costs. In addition to the overall high quality of the leading FOSS software, they discovered additional advantages to FOSS during their evaluation processes. First, they did not have to engage with a vendor to download and use the software. If they were evaluating FOSS software for which commercial support services were offered, they could wait until they were ready to deploy the software for a business-critical application before spending any money. Next, companies could choose to use a community version of the software for less critical internal needs even as they used a commercial version for critical ones. For example, a company wishing to use a FOSS content management system could download the noncommercial version of Drupal, as well as a commercially supported Drupal distribution from Acquia.

The power of the Internet to support FOSS development has thus transformed the software industry in several ways beyond those noted in Sect. 2. First, some proprietary companies have changed their business models or introduced new products as a new way to compete with FOSS software. For example, two of the leading vendors of commercial relational DBMSs (Oracle and IBM) have introduced free versions of their products. These free versions have some limitations in their capacity, but are otherwise identical to their mainline commercial products.

These vendors made the strategic decision that it was preferable to give away a version of their product in the hope that the end user's needs would grow to the point where they would upgrade from the free version to the full product, or pay for product support for the free product. The alternative would have been to lose these users, possibly forever, to a FOSS product. It is highly unlikely that these leading software vendors would have taken that approach if it were not for the easily available open source alternative.

Many traditional proprietary vendors of software have long relied on a business model dominated by the sale of product licenses with additional revenue from annual maintenance fees and professional services. In some cases, though, the high quality of the open source alternative has captured a large share of the market for such products. As just one example, Borland lost a huge percentage of its market share for C++ and Java programming environments to the open source Eclipse environment and its ecosystem of plug-in extensions. JetBrains, developer of IntelliJ, introduced an open source version of their IDEA Java development environment, cannibalizing sales from their main commercial product.

Even Microsoft, the largest software vendor, is affected by this transformation, as customers have been slow to upgrade to recent versions of Microsoft Office or have migrated to a hosted or open source alternative. Microsoft's competitive response has included creation of a less expensive and less feature-rich Home and Student edition of Office, as well as development of a hosted service, live.com, for editing of documents compliant with the Office file formats. In summary, many vendors of proprietary software products have found their markets and business models disrupted not just by the Internet, but also by free and open source software projects and companies that have taken advantage of the Internet to produce compelling alternatives at drastically lower prices [28].

4 Looking ahead

As this paper has shown, the advent of the Internet has had a tremendous impact on the software industry. In response, existing companies have modified their software products, their development methods, and their practices for sales, marketing, and support. The Internet has enabled companies to reduce their costs for sales, marketing, and support. At the same time, many of these companies have been challenged by newer companies with different business approaches, including hosted applications and open source software.

The software industry has previously been disrupted by the transitions from mainframes to minicomputers, then from minicomputers to personal computers, and then by local networking that enabled client-server applications. It has been similarly disrupted by changes in user interfaces, particularly the change from alphanumeric displays to graphical displays and windowing systems. Each wave of technological change brings new companies into the market, often displacing incumbents. For example, the companies that were considered industry leaders in the first round of database management systems (DBMSs) were displaced by the vendors of relational DBMSs. Today's database products include nonrelational alternatives [30], and these may

eventually gain a significant market share in the transition to cloud computing.

Vint Cerf, one of the "fathers" of the Internet, has pointed to the "Internet of things" and speech recognition, among other things, as important aspects of future web development [4]. Those advances will undoubtedly lead to new and more powerful applications, as well as having an impact on today's applications.

Mobile applications, or apps, have already had a major impact on the software industry. Not only have thousands of small software companies emerged to lead the development of these apps, but the price of these apps is typically two orders of magnitude lower than that of software for personal computers, if not free.

Today's mobile devices use a widget, touch, and motion style of interaction, departing from the WIMP (Windows, Icons, Menus, Pointer) style of user interaction dominant for more than two decades. Many basic features, including calling or texting a personal contact, or creating a route to a destination, have now been voice-enabled, further reducing the need for either a physical or virtual keyboard, and promising another transition in user interaction.

Limited storage capacities on mobile devices have also changed the architecture of applications, where many mobile web applications have very limited functionality on the mobile device, delivering functionality and content from the server side. As one example, Amazon.com has recently introduced an application that allows people to store music on Amazon.com servers (for a nominal cost) and deliver it on request, rather than storing the music on the mobile device itself.

This last example shows the logical progression of a long-term trend. In the early days of computing, all computing and storage was restricted to a single machine. The development of local networks enabled separation of computing and storage, as well as breaking applications into separate components. In the early days of the Internet, the FTP and Gopher protocols supported remote access to data. The emergence of the World Wide Web made it possible to execute applications locally and remotely, delivering content to a local device. Large scale consumer-oriented web applications are often architected in a way that makes it impossible for an average user to tell where the application is executing and where the content is stored; in many cases, content is replicated globally and delivered as needed through a content delivery network.

Today, organizations are moving to a cloud computing model for their own needs. Rather than owning their own servers and paying for underutilized resources, they are moving toward public and private clouds for their computing needs, paying only for the resources that they actually are using. Amazon Web Services, as a typical example, allows customers to configure their own virtual machines, to host their own applications, to store their own data, and to dynamically scale up or down their resources as needed [2]. Public clouds are effectively utilities, available to anyone wishing to pay for the service; this idea in not new, but was envisioned more than 40 years ago [20].

In summary, the growth in mobile applications, changes in user interaction, availability of location-based information, and the continued growth of cloud computing are all technology-based indications that applications and the software industry will continue to change. It is not hard to imagine entirely new categories of applications, such as biosensor-based diagnostic applications in health care and disaster prediction software that will draw on these trends.

Beyond the technological changes, product pricing and software industry consolidation are likely to affect the software industry. As noted above, the cost for mobile applications is much lower than the cost of traditional applications. In addition to "smartphones," tablets, such as Apple's iPad, continue to take market share from notebook and desktop computers, and have become the fastest-growing category of computers. If these low prices for applications becomes the norm, that could again disrupt the software industry. Rovio, the maker of the Angry Birds mobile game, currently derives millions of dollars in revenue from a high volume of inexpensive licenses, plus in-game advertising. That model is fundamentally different from those of traditional software companies.

In addition, the software industry has been consolidating for many years, with established companies continuing to make strategic acquisitions. There are many reasons why these acquisitions have occurred and are likely to continue. First, it is very difficult and expensive for a software company to make the transition from a private company to a publicly owned company; throughout the twenty-first century, it has been much easier for a company to sell itself to a public company as a way for its investors, founders, and employees to make some money from their efforts and investments. Second, there are economies of scale that make it more efficient, i.e., more profitable, to combine smaller software companies into a large one. Third, the threats of disruptive innovations and technological changes present a continuing challenge to every software company, even today's industry leaders. It is often in their best interests to join forces with a larger company rather than to lose the value of their business. Thus, Sybase, which had previously acquired Powersoft (and other companies), recently agreed to be acquired by SAP.

In conclusion, then, the Internet has brought vast changes to the software industry. Its long term impact is difficult to predict, since the technology is so new. Even now, however, the Internet has become pervasive in the lives of billions of people and has transformed the way that people live, work,

and communicate. It is certain that future technological advances and business considerations will continue to affect the entire world.

Acknowledgements I am most grateful to the Editors-in-Chief, Fabio Kon and Gordon Blair, for giving me the opportunity to write this paper. They provided valuable comments and suggestions on its earlier drafts.

References

1. Apache Software Foundation (2010) Apache HTTP Server 2.2 Official Documentation—Volume I. Server Administration. Fultus
2. Barr J (2010) Host your web site in the cloud: Amazon web services made easy. SitePoint Pty, Melbourne
3. Capra E, Wasserman AI (2008) A framework for evaluating managerial styles in open source projects. In: Proc. 4th int'l conference on open source systems, pp 1–11
4. Cerf V (2010) A half-century makes a difference. J Internet Serv Appl 1(1):3–5
5. DeLoura M (2009) The engine survey: general results. http://www.satori.org/2009/03/the-engine-survey-general-results. Accessed on 7 April 2011
6. Flanagan D, Matsumoto Y (2008) The Ruby programming language. O'Reilly, Sebastopol
7. Gay J, Stallman RM (2009) Free software, free society: selected essays of Richard M. Stallman. CreateSpace
8. Goldberg A (1983) Smalltalk-80: the interactive programming environment. Addison-Wesley, Reading
9. Halligan B, Shah D (2009) Inbound marketing: get found using Google, social media, and blogs. Wiley, New York
10. Ierusalimschy R (2006) Programming in Lua, 2nd edn. Lua.org
11. Johnson R et al. (2005) Professional Java development with the Spring Framework. Wrox
12. Kaushik A (2009) Web Analytics 2.0: the art of online accountability and science of customer centricity. Sybex, Indianapolis
13. Kemeny JG, Kurtz TE (1985) Back to BASIC: the history, corruption, and future of the language. Addison-Wesley, Reading
14. Kernighan BW, Mashey JR (1979) The Unix programming environment. Softw Pract Exp 9(1):1–15
15. Kushner D (2004) Masters of doom: how two guys created an empire and transformed pop culture. Random House, New York
16. Lerdorf R, Tatroe K, McIntyre P (2006) Programming PHP. O'Reilly, Sebastopol
17. Lutz M (2010) Programming Python, 4th edn. O'Reilly, Sebastopol
18. Netscape. Wikipedia entry. http://en.wikipedia.org/wiki/Netscape. Accessed on 7 April 2011
19. Open Source Initiative (1998) The Open Source Definition. http://opensource.org/docs/osd. Accessed on 7 April 2011
20. Parkhill DF (1966) The challenge of the computer utility. Addison-Wesley, Reading
21. Pope K (2009) Zend framework 1.8 web application development. Packt, Birmingham
22. Rochkind M (1975) The source code control system. IEEE Trans Softw Eng SE-1(4):364–370
23. Robinson D, Coar K (2004) The Common Gateway Interface (CGI) Version 1.1. http://www.ietf.org/rfc/rfc3875. Accessed on 6 April 2011
24. Ruby S, Thomas D, Hansson DH (2011) Agile web development with Rails, 4th edn. Pragmatic Bookshelf, Lewisville
25. Schwartz R, Phoenix T, Foy B (2008) Learning Perl, 5th edn. O'Reilly, Sebastopol
26. Skim PDF Reader and Note-taker for OS X. http://sourceforge.net/projects/skim-app/files/Skim/. Accessed on 13 March 2011
27. Van Rossum G, Drake FL Jr (2003) The Python language reference manual. Network Theory
28. Wasserman AI (2009) Building a business on open source software. In: Petti C (ed) Cases in technological entrepreneurship: converting ideas into value. Edward Elgar, Chaltenham Glos, pp 107–121
29. Wasserman AI, Pircher PA (1987) A graphical extensible integrated environment for software development. ACM SIGPLAN Not 22(1):131–142 (Proceedings of the 2nd ACM SIGSOFT/SIGPLAN software engineering symposium on Practical software development environments)
30. White D (2010) Hadoop: the definitive guide. O'Reilly, Sebastopol