Understanding Software: A Primer for Managers

Last modified: March 8, 2009 (draft version: comments very much welcomed)

INTRODUCTION

We know computing hardware is getting faster and cheaper, creating all sorts of exciting and disruptive opportunities for the savvy manager. But what's really going on inside the box? It's *software* that makes the magic of computing happen. Without software, your PC would be a heap of silicon, wrapped in wires encased in plastic and metal. But it's the instructions—the software code--that enable a computer to do something wonderful, driving the limitless possibilities of information technology.

Software is everywhere. An inexpensive cell phone has about 1 million lines of code, while the average car contains nearly 100 million[1]. In this chapter we'll take a peek inside the chips to understand what software is. There are a lot of terms associated with software: operating systems, applications, enterprise software, distributed systems, and more. We'll define these terms up front, and put them in a managerial context . A follow-up chapter, "Software in Flux", will focus on changes impacting the software business, including open source software, software as a service (SaaS), and cloud computing. These changes are creating an environment radically different from the software industry that existed in prior decades – confronting managers with a whole new set of opportunities and challenges.

Managers who understand software can better understand the possibilities and impact of technology. They can make better decisions regarding the strategic value of IT, and the potential for technology-driven savings. They can appreciate the challenges, costs, security vulnerabilities, legal and compliance issues, and limitations involved in developing and deploying technology solutions. In the next two chapters we will closely examine the software industry and discuss trends, developments and economics – all of which influence decisions managers make about products to select, firms to partner with, and firms to invest in.

WHAT IS SOFTWARE

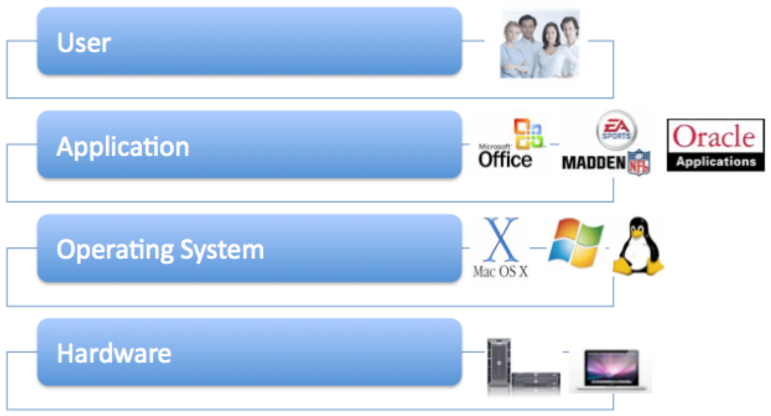
When we refer to *computer hardware* (sometimes just hardware), we're talking about the physical components of information technology – the equipment that you can physically touch. This includes computers, storage devices, networking equipment, and other peripherals.

Software refers to a computer program or collection of programs – sets of instructions that tell the hardware what to do. Software gets your computer to behave like a web browser or word processor, makes your iPod play music and video, and enables your bank's ATM to spit out cash.

[1] Charette, 2005

It's when we start to talk about the categories of software that most people's eyes glaze over. To most folks, software is a big, incomprehensible alphabet soup of acronyms and geeky phrases: OS, VB, SAP, SQL, to name just a few.

Don't be intimidated. The basics are actually pretty easy to understand. But it's not soup, it's more of a layer-cake. Think about computer hardware as being at the bottom of the layer cake. The next layer is the *operating system* – the collection of programs that control the hardware. Windows, Mac OS X, and Linux are operating systems. On top of that layer are *applications* – these can range from end-user programs like those in Office, to the complex set of programs that manage a business's inventory, payroll, and accounting. At the top of the cake are users.



The Hardware/Software 'Layer Cake' with the Users at the Top

The flexibility of these layers gives computers the customization options that managers and businesses demand. Understanding how the layers relate to each other helps you make better decisions on what options are important to your unique business needs, can influence what you buy, and may have implications for everything from competitiveness to cost overruns to security breaches. What follows is a manager's guide to the main software categories, with an emphasis on why each is important.

KEY TAKEAWAYS

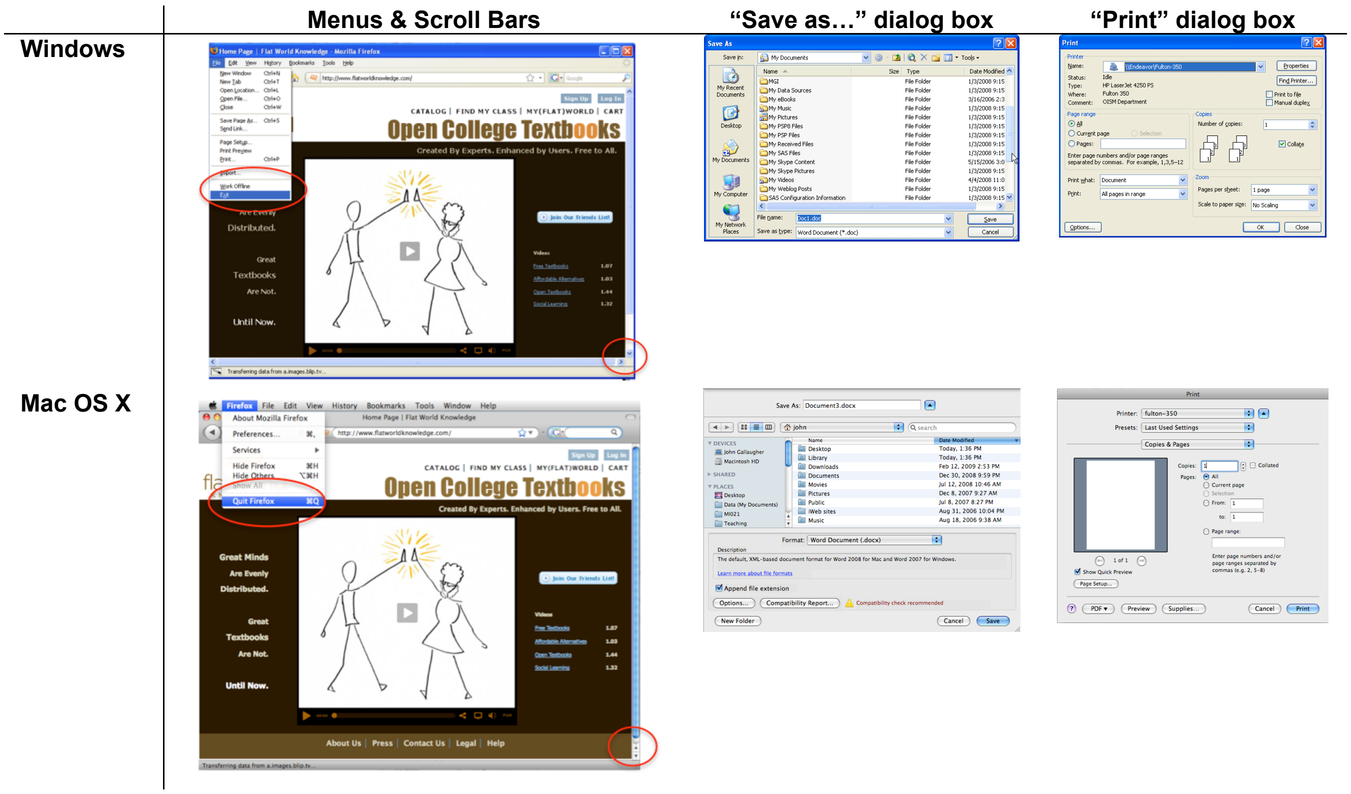
- Software refers to a computer program or collection of programs.
- You can think of software as being part of a layer cake, with hardware at the bottom; the operating system controlling the hardware, the applications executing one layer up, within rules and constraints established by the operating system; all serving users and organizations.
- How these layers relate to one another has managerial implications in many areas, including the flexibility in meeting business demand, costs, legal issues and security.

OPERATING SYSTEMS

Computing hardware needs to be controlled, and that's the role of the *operating system*. The operating system (sometimes called the *OS*) provides a common set of controls for managing computer hardware, making it easier for users to interact with computers and for programmers to write application software.

Anyone who has used a PC or a Mac has used an operating system. And anyone who has used them both and noticed differences across these platforms can get a sense of the breadth of what an operating system does. Even for programs that are otherwise identical for these two systems (like the Firefox browser), subtitle differences are visible. Screen elements like menus, scroll bars, and window borders look different on the Mac than they do in Windows. So do the dialog boxes that show up when you print or save.

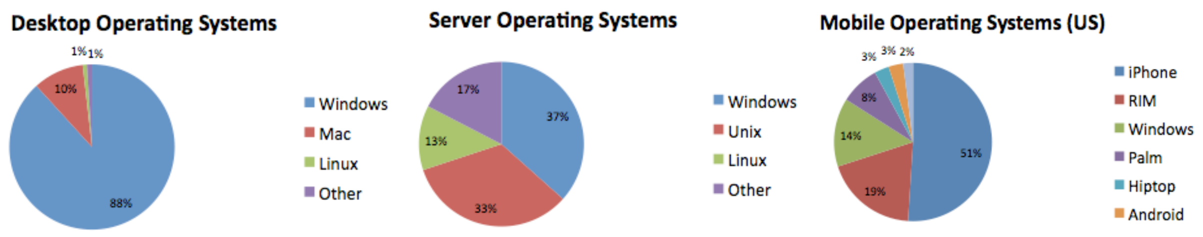
These items look and behave differently because each of these functions touches the hardware, and the team that developed Microsoft Windows created a system distinctly different from their Macintosh counterparts at Apple. *User interface* items like scroll bars and menus are displayed on the hardware of the computer display. Files are saved to the hardware of a hard drive or other storage device. Most operating systems also include control panels, desktop file management, and other support programs to work directly with hardware elements like storage devices, displays, printers, and networking equipment. The Macintosh Finder or the Windows Explorer, are examples of components of these operating systems. See the figure for similarities and differences.



Differences between the Windows and Mac operating systems are evident throughout the user interface, particularly when a program interacts with hardware (source: Gallaugher)

Operating systems are also designed to give programmers a common set of commands to consistently interact with the hardware. This makes a programmer's job easier; it reduces program complexity (and chance for errors in code). It also helps to create a consistency in look, feel, and functionality across all of the programs offered for an OS – this makes it easier to learn new software, and reduces training costs.

Just about every computing device has an operating system – desktops and laptops, enterprise-class server computers, your mobile phone. Even specialty devices like iPods and video game consoles have their own OS.



Operating system market share for desktop ('09), server ('08), and mobile phones ('09)
Data: HitsLink, Forrester, IDC, and admob

A common operating system means a less complex task for software developers. Think about the advantage to the video game programmer – why should each coder write their own separate routines to access the Wiimote, play sound, draw graphics, or save files when Nintendo can give every game developer a simple command to use consistently in all programs?

Firmware & Embedded Systems

Most personal computers have an operating system installed on their hard drives. This allows the OS to be replaced or upgraded easily. But many smaller, special purpose computing devices have their operating systems installed on non-volatile memory (often on ROM or read-only memory chips). Control programs stored on chips are sometimes referred to as *firmware*. The OS in an iPod, mobile phone, or your TV's set-top box is most likely stored as firmware. Your PC also has a tiny bit of firmware that allows it to do very basic functions like startup (*boot*) and begin loading its operating system from disk.

Another term you might hear is *embedded systems*. As computing gets cheaper, special-purpose technology is increasingly becoming embedded into all sorts of devices like cars, picture frames, aircraft engines, photocopiers, and heating and air conditioning systems. The software programs that make up embedded systems are often stored as firmware, too.

Moore's law (see the chapter "*Moore's Law and More*") enables embedded systems, and these systems can create real strategic value. The Otis Elevator Company, a division of United Technologies, uses embedded systems in its products to warn its service centers when the firm's elevators, escalators, and moving walkways need maintenance or repair. This provides Otis with several key benefits: 1) Since products automatically contact Otis when they need attention, these systems generate a lucrative service business for the firm, and make it more difficult for third parties to offer a competing business servicing Otis products; 2) Products contact service technicians to perform maintenance based on exact needs (e.g. lubricant is low, or a part has been used enough to be replaced) rather than guessed schedules. This makes service more cost effective, products break down less, and customers are happier; 3) Any product failures are immediately detected, with embedded systems typically dispatching technicians before a client's phone call; and 4) The data is fed back to Otis's R&D group, providing information on reliability and failure so that engineers can use this info to design better products. These are pretty big benefits from software embedded on tiny chips.

KEY TAKEAWAYS

- The operating system (OS) controls a computer's hardware and provides a common set of commands for writing programs.
- Just about every general-purpose computing device (PC, phone, set top box) has an operating system.
- Embedded systems are special-purpose computer systems designed to perform one or a few dedicated functions, and are frequently built into conventional products like cars, air conditioners, and elevators.
- Embedded systems can enable entire new businesses and offer firms resources to build competitive advantage.

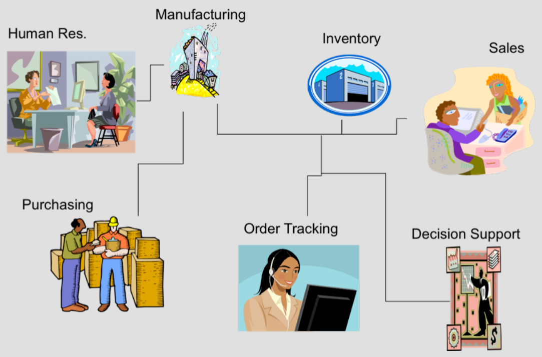
APPLICATION SOFTWARE

Operating systems are designed to create a *platform* so that programmers can write additional applications, allowing the computer to do even more useful things. While operating systems control the hardware, *application software* (sometimes referred to as *software applications*, *applications*, or even just *apps*) perform the work that users and firms are directly interested in accomplishing. Think of applications as the place the user's or organization's real work gets done. As we've learned in our *network effects* chapter, the more specialized application software that is available for a platform (the more games for a video game console, the more apps for your phone), the more valuable it potentially becomes.

Desktop software refers to applications installed on a personal computer. Think your browser, your Office suite (e.g. word processor, spreadsheet, presentation software), photo editors, computer games, etc. *Enterprise software* refers to applications that address the needs of multiple, simultaneous users in an organization or work group. Most companies run various forms of enterprise software programs to keep track of their inventory, record sales, manage payments to suppliers, cut employee paychecks, and handle other functions.

Some firms write their own enterprise software from scratch, but this can be time consuming and costly. Since many firms have similar procedures for accounting, finance, inventory management, and human resource functions, it often makes sense to buy a *software package* (a software product offered commercially by a third party) to support these functions. So-called *enterprise resource planning* (or *ERP*) software packages serve precisely this purpose. In the way that Microsoft can sell you a suite of desktop software that works together, many companies sell ERP software that integrates the functions of a business. The leading ERP vendors include the firm's SAP and Oracle, although there are many firms that sell ERP software. A company doesn't have to install all of the modules of an ERP suite, but it might add functions over time – say to plug in an accounting program that is able to read data from the firms previously installed inventory management system. And although a bit more of a challenge to integrate, a firm can also mix and match components, linking software the firm has written with modules purchased from different enterprise software vendors.

ERP in Action[2]



An ERP system with multiple modules installed can touch many functions of the business:

- SALES - A sales rep from Vt.-based SnowboardCo. takes an order for 5,000 boards from a French sporting goods chain. The system can verify credit history, apply discounts, and calculate price (in Euros)
- INVENTORY - While the sales rep is on the phone with his French customer, the system immediately checks product availability, signaling that 1,000 boards are ready to be shipped from the firm's Montpellier warehouse, the other 4,000 need to be manufactured and can be delivered in two weeks from the firm's manufacturing facility in Guangzhou.
- MANUFACTURING - When the customer confirms the order, the invoice is printed in French, priced in Euros, and the system notifies the Guangzhou factory to ramp up production for the model ordered.
- HUMAN RESOURCES - High demand across this weeks' orders triggers a notice to the Guangzhou hiring manager, notifying her that the firm's products are a hit, and that the flood of orders coming in globally mean her factory will have to hire 5 more workers to keep up.
- PURCHASING - The system keeps track of raw material inventories, too. New orders trigger an automatic order with SnowboardCo's suppliers, so that raw materials are on hand to meet demand.
- ORDER TRACKING - The French customer can log in to track her SnowboardCo order. The system shows her other products that are available, using this as an opportunity to cross-sell additional products.
- DECISION SUPPORT - Management sees the firm's European business is booming, and plans a marketing blitz for the continent, targeting board models that seem to sell better for the Alps crowd than in the US Market.

Other categories of enterprise software that managers are likely to encounter include:

- *Customer Relationship Management* (*CRM*) systems, used to support customer-related sales and marketing activities;
- *Supply Chain Management* (*SCM*) systems, that can help a firm manage aspects of its value chain, from the flow of raw materials into the firm, through delivery of finished products and services at the point-of-consumption; and
- *Business Intelligence* (*BI*) systems, which use data created by other systems to provide reporting and analysis for organizational decision making.

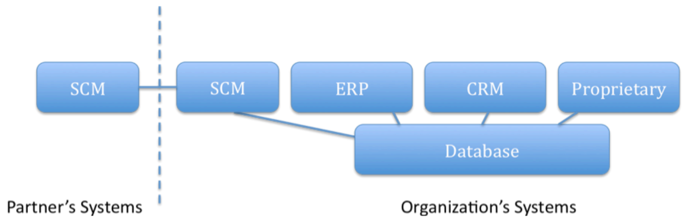
Major ERP vendors are now providing products that extend into these and other categories of enterprise application software, as well.

Most enterprise software works in conjunction with *database management software* (sometimes referred to as a *DBMS* or *database system*). The database system stores and retrieves the data that an application creates and uses. Think of this as another additional layer in our cake analogy. The database system sits above the operating system, but under the enterprise applications. Many ERP systems and enterprise software programs are configured to share the same database system so that an organization's different programs can use a common, shared set of data. This can be hugely valuable for a company's efficiency. For example, this could allow a separate set of programs that manage an inventory and point-of-sale system to update a single set of data that tells how many products a firm has to sell, and how many it has already sold –

[2] Example loosely based on Edmondson, 1997; graphics from BusinessWeek

information that would also be used by the firm's accounting and finance systems to create reports showing the firm's sales and profits.

Firms that don't have common database systems with consistent formats across their enterprise often struggle to efficiently manage their value chain. Common procedures and data formats created by packaged ERP systems and other categories of enterprise software also make it easier for firms to use software to coordinate programs between organizations. This can lead to even more value chain efficiencies. Sell a product? Deduct it from your inventory. When inventory levels get too low, have your computer systems send a message to your supplier's systems so that they can automatically build and ship replacement product to your firm. In many cases these messages are sent without any human interaction, reducing time and errors.



An organization's database management system can be set up to work with several applications both within and outside the firm (source: Gallaugher)

The Rewards and Risks of Packaged Enterprise Systems

When set up properly, enterprise systems can save millions of dollars and turbo-charge organizations. For example, the CIO of office equipment maker Steelcase credited the firm's ERP with an $80 million reduction in operating expenses saved from eliminating redundant processes and making data more usable. The CIO of Colgate Palmolive also praised their ERP, saying "The day we turned the switch on, we dropped two days out of our order-to-delivery cycle."[3] Packaged enterprise systems can streamline processes, make data more usable, and ease the linking of systems with software across the firm and with key business partners. Plus, the software that makes up these systems is often debugged, tested, and documented with an industrial rigor that may be difficult to match with proprietary software developed in-house.

But for all the promise of packaged solutions for standard business functions, enterprise software installations have proven difficult, standardizing business processes in software that others can buy means that those functions are easy for competitors to match, and the vision of a single monolithic system that delivers up wondrous efficiencies has been difficult for many to achieve. The average large company spends roughly $15 million on ERP software, with some installations running into the hundreds of millions of dollars[4]. And many of these efforts have failed disastrously.

FoxMeyer was once a $6 billion drug distributor, but a failed ERP installation led to a series of losses that bankrupted the firm. The collapse was so rapid and so complete that just a year after launching the system, the carcass of what remained of the firm was sold to a rival for less than $80 million. Hershey Foods blamed a $466 million revenue shortfall on glitches in the firm's ERP rollout. Among the problems, the botched implementation prevented the candy maker from getting product to stores during

[3] Robinson and Dilts, 1999
[4] Rettig, 2007

the critical period before Halloween. Nike's first SCM and ERP implementation was labeled a 'disaster', their systems were blamed for over $100 million in lost sales[5]. Even tech firms aren't immune to software implementation blunders. HP once blamed a $160 million loss on problems with its ERP systems[6]. Manager beware – there are no silver bullets. For insight on the causes of massive software failures, and methods to improve the likelihood of success, see the section '*Why Software Fails*' later in this chapter.

KEY TAKEAWAYS

- Applications software focuses on the work of a user or an organization.
- Desktop applications are designed for a single user. Enterprise software supports multiple users in an organization or work group.
- Popular categories of enterprise software include ERP (enterprise resource planning), SCM (supply chain management), CRM (customer relationship management), and BI (business intelligence) software, among many others.
- These systems are used in conjunction with database management systems, programs that help firms organize, store, retrieve, and maintain data.
- ERP and other packaged enterprise systems can be challenging and costly to implement, but can help firms create a standard set of procedures and data that can ultimately lower costs and streamline operations.

DISTRIBUTED COMPUTING

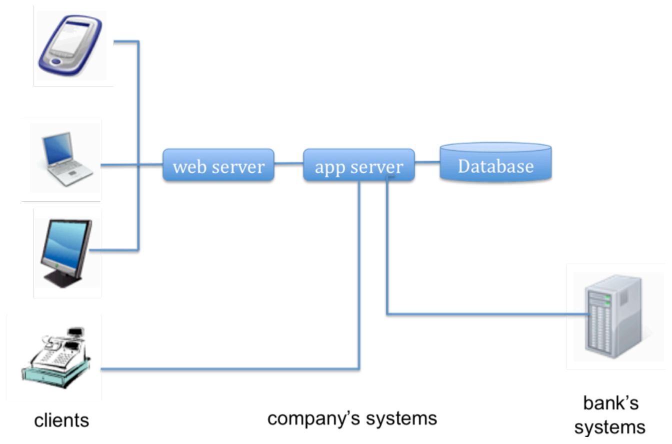
When computers in different locations can communicate with one another, this is often referred to as *distributed computing*. Distributed computing can yield enormous efficiencies in speed, error reduction, and cost savings and can create entirely new ways of doing business. Designing systems architecture for distributed systems involves many advanced technical topics. The examples that follow will help managers understand the bigger ideas behind some of the terms they're likely to encounter.

Let's start with the term *server*. This is a tricky one because it's frequently used in two ways, 1) in a hardware context a server is a computer that has been configured to support requests from other computers (e.g. Dell sells servers) and, 2) in a software context a server is a program that fulfills requests (e.g. the Apache open source web server). Most of the time, server *software* resides on server-class *hardware*, but you can also set up a PC, laptop, or other small computer to run server software, albeit less powerfully. And you can use mainframe or super-computer class machines as servers, too.

The World Wide Web, like many other distributed computing services, is what geeks call a *client-server* system. Client-server refers to two pieces of software, a *client* that makes a request, and a *server* that receives and attempts to fulfill the request. In our WWW scenario, the client is the browser (e.g. Internet Explorer, Firefox, Safari). When you type a web page's address into the location field of your browser, you're telling the client to 'go find the web server software at the address provided, and tell the server to return the web page requested'.

[5] Koch, 2004
[6] Charette, 2005

It is possible to link simple scripting languages to a web server for performing calculations, accessing databases, or customizing web pages. But more advanced distributed environments may use a category of software called an *application server*. The application server (or app server) houses business logic for a distributed system. Individual *web services* served up by the app server are programmed to perform different tasks. This could include: returning a calculation ("sales tax for your order will be $11.58"), accessing a database program ("here are the results you searched for"), or even making a request to another server in another organization ("Visa, please verify this customer's credit card number for me").



In this multi-tiered distributed system, clients' browsers on various machines (desktop, laptop, mobile) access the system through the web server. The cash register doesn't use a web browser, so instead the cash register logic is programmed to directly accesses the services it needs from the app server. Web services accessed from the app server may be asked to do a variety of functions, including perform calculations, access corporate databases, or even make requests from servers at other firms (for example, to verify a customer's credit card)
(source: Gallaugher)

Those little chunks of code that are accessed via the application server are sometimes referred to as web services. The World Wide Web consortium defines *web services* as software systems designed to support interoperable machine-to-machine interaction over a network[7]. And when computers can talk together (instead of people), this often results in fewer errors, time savings, cost reductions, and can even create whole new ways of doing business! Each web service defines the standard method for other programs to request it to perform a task, and defines the kind of response the calling client can expect back. These standards are referred to as *APIs* or application programming interfaces.

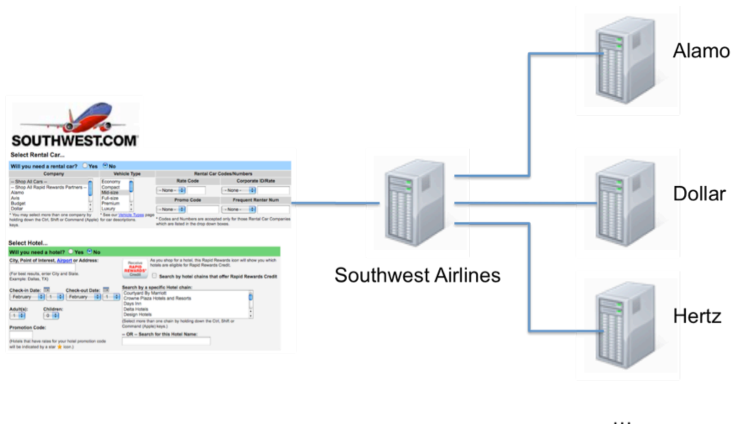
Look at the advantages that web services bring a firm like Amazon. Using web services, the firm can allow the same order entry logic to be used by web browsers, mobile phone applications, or even by third parties who want to access Amazon product information and place orders (there's an incentive to funnel sales to Amazon – the firm will give you a cut of any sales). Organizations that have created a robust set of web services around their processes and procedures are said to have a *service oriented architecture*, or *SOA*. Organizing systems like this, with separate applications in charge of client presentation, business logic, and database, makes systems more flexible. Code can be reused, and each layer can be separately maintained,

[7] W3C, 2004

upgraded, or migrated to new hardware – all with little impact on the others.

Web services sound geeky, but here's a concrete example illustrating their power: Southwest Airlines had a website where customers could book flights, but many customers also wanted to rent a car or book a hotel, too. To keep customers on Southwest.com, the firm and its hotel and rental car partners created a set of web services and shared the APIs. Now customers visiting Southwest.com can book a hotel stay and rental car on the same page that they make their flight reservation. This transforms Southwest.com into a full service travel destination and allows the site to compete head-to-head with the likes of Expedia, Travelocity, and Orbitz[8].

Think about why web services are important from a strategic perspective. By adding hotel and rental car services, Southwest is now able to eliminate the travel agent, along with any fees they might share with the agent. This allows the firm to capture more profits or pass on savings to customers, securing its position as the first place customers go for low-cost travel. And perhaps most importantly, Southwest can capture key data from visitor travel searches and bookings (something it likely couldn't do if customers went to a site like Expedia or Travelocity, instead). This kind of customer data can be used by Southwest to send out custom e-mail messages and other marketing campaigns to bring customers back to the airline. As geeky as they might at first seem, web services can be very strategic!



Southwest.com uses web services to allow car rental and hotel firms to book services through Southwest. This transforms Southwest.com into a full-service online travel agent (source: Gallaugher)

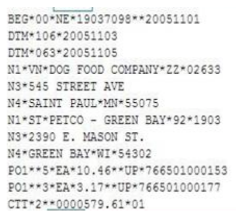
Messaging Standards

Two additional terms you might hear within the context of distributed computing EDI and XML. *EDI* or *Electronic Data Interchange* is a set of standards for exchanging information between computer applications. EDI is most often used as a way to send the electronic equivalent of structured documents between different organizations. Using EDI, each element in the electronic document, like a firm name, address, or customer number, is coded so that it can be recognized by the receiving computer program. Eliminating paper documents makes businesses faster and

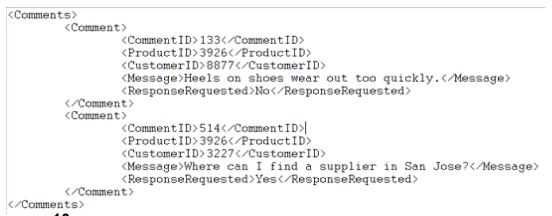
[8] McCarthy, 2002

lowers data entry and error costs. One study showed that firms that used EDI decreased their error rates by 82% and their cost of producing each document fell by up to 96%[9].

EDI is a very old standard, with roots stretching back to the 1948 Berlin Air Lift. While still in use, a new generation of more flexible technologies for specifying data standards are taking its place. Chief among the technologies replacing EDI is *XML*, which stands for *eXtensible Markup Language*. XML has lots of uses, but in the context of distributed systems, it allows software developers to create a set of standards for common data elements that, like EDI messages, can be sent between different kinds of computers, different applications, and different organizations. XML is often thought of as easier to code than EDI, and it's more robust because it can be extended – organizations can create formats to represent any kind of data (e.g. a common part number, photos, the complaint field collected by customer support personal). In fact, most messages sent between web services are coded in XML. Many computer programs also use XML as a way to export and import data in a common format that can be used regardless of the kind of computer hardware, operating system, or application program used. And if you design web pages, you might encounter XML as part of the coding behind the cascading style sheets (CSS) that help maintain a consistent look and feel to the various web pages in a given website.



```
BEG*00*NE*19037098**20051101
DTM*106*20051103
DTM*063*20051105
N1*VN*DOG FOOD COMPANY*ZZ*02633
N3*545 STREET AVE
N4*SAINT PAUL*MN*55075
N1*ST*PETCO - GREEN BAY*92*1903
N3*2390 E. MASON ST.
N4*GREEN BAY*WI*54302
PO1**5*EA*10.46**UP*766501000153
PO1**3*EA*3.17**UP*766501000177
CTT*2**0000579.61*01
```



```
<Comments>
        <Comment>
                <CommentID>133</CommentID>
                <ProductID>3926</ProductID>
                <CustomerID>8877</CustomerID>
                <Message>Heels on shoes wear out too quickly.</Message>
                <ResponseRequested>No</ResponseRequested>
        </Comment>
        <Comment>
                <CommentID>514</CommentID>
                <ProductID>3926</ProductID>
                <CustomerID>3227</CustomerID>
                <Message>Where can I find a supplier in San Jose?</Message>
                <ResponseRequested>Yes</ResponseRequested>
        </Comment>
</Comments>
```

Left: portion of a document encoded in EDI[10]. Right: portions of a different document coded in XML[11]
Source not approved

Rearden Commerce: A Business Built on Web Services

Web services, APIs, and open standards not only transform businesses, they can create entire new firms that change how we get things done. For a look at the mashed-up, integrated, hyper-automated possibilities that web services make possible, check out Rearden Commerce, a Foster City, CA firm that is using this technology to become what AMR's Chief Research Office referred to as 'Travelocity on Steroids'.

Using Rearden, firms can offer their busy employees a sort of web-based concierge/personal assistant. Rearden offers firms a one-stop shop where employees can not only make the flight, car, and hotel bookings they might do from a travel agent, they can also book dinner reservations, sports and theatre tickets, and arrange for business services like conference calls and package shipping. Rearden doesn't supply the goods and services it sells. Instead it acts as the middleman between transactions. A set of open APIs to its web services allows Rearden's 160,000 suppliers to send product and service data to Rearden, and to receive booking and sales data from the site.

[9] National Petroleum News, 1998
[10] http://www.ec-bp.biz/joomla/images/stories/siteimages/isa-ieaenv.jpg
[11] http://www.ibm.com/developerworks/data/library/techarticle/dm-0603saracco2/CommentXML.gif

In this ultimate business mashup, a mobile Rearden user could use her phone to book a flight into a client city, see restaurants within a certain distance of her client's office, have these locations pop-up on a Google map, have listings accompanied by Zagat ratings and cuisine type, book restaurant reservations through Open Table, arrange for a car and driver to meet her at her client's office at a specific time, and sync up these reservations with her firm's corporate calendaring systems. If something unexpected comes up, like a flight delay, Rearden will be sure she gets the message. The system will keep track of any cancelled reservation credits, and also records travel reward programs, so Rearden can be used to spend those points in the future.

In order to pull off this effort, the Rearden maestros are not only skilled at technical orchestration, but also in coordinating customer and supplier requirements. As TechCrunch's Erick Schonfeld put it "*The hard part is not only the technology—which is all about integrating an unruly mess of APIs and Web services—[it also involves] signing commercially binding service level agreements with [now over 160,000] merchants across the world*". For its efforts, Rearden gets to keep between 6% and 25% of every non-travel dollar spent, depending on the service. The firm also makes money from subscriptions, and distribution deals.

The firm's first customers were large businesses and included ConAgra, GlaxoSmithKline, and Motorola. Rearden's customers can configure the system around special parameters unique to each firm to favor a specific airline, benefit from a corporate discount, or to restrict some offerings only to approved employees. Rearden investors include JP Morgan Chase and American Express – both of whom offer Rearden to their employees and customers. Even before the consumer version was available, Rearden had over 4,000 corporate customers and 2 million total users, a user base larger than better-known firms like Salesforce.com[12].

Connectivity has made our systems more productive, and enables entire new strategies and business models. But these wonderful benefits come at the price of increased risk. When systems are more interconnected, opportunities for infiltration and abuse also increase. Think of it this way – each 'connection' opportunity is like adding another door to a building. The more doors that have to be defended, the more difficult security becomes. It should be no surprise that the rise of the Internet and distributed computing has led to an explosion in security losses by organizations worldwide. For more on this, see the *Security* chapter.

KEY TAKEAWAYS

- Client-server computing is a method of distributed computing where one program (a client) makes a request to be fulfilled by another program (a server).
- Web servers serve up web pages and can perform some scripting.
- Most firms serve complex business logic from an application server.
- Isolating a system's logic in three or more layers (presentation or user interface, business logic, and database) can allow a firm flexibility in maintenance, reusability, and in handling upgrades.
- Web services allow different applications to communicate with one another. APIs define the method to call a web service (e.g. to get it to do something), and the kind of response the calling program can expect back.
- Web services make it easier to link applications as distributed systems, and can make it easier for firms to link their systems across organizations.

[12] Arrington, 2007; Schofield, 2008; Arrington, 2009

- Popular messaging standards include EDI (older) and XML. Sending messages between machines instead of physical documents can speed processes, drastically cut the cost of transactions, and reduce errors.

WRITING SOFTWARE

So you've got a great idea that you want to express in software – how do you go about creating a program? Programmers write software in a programming language. While each language has its strengths and weaknesses, most commercial software is written in C++ (pronounced 'see plus plus') or C# (pronounced 'see sharp'). Visual Basic (from Microsoft) and Java (from Sun) are also among the more popular of the dozens of programming languages available.

Most professional programmers use an *integrated development environment (IDE)* to write their code. The IDE includes a text editor, a debugger for sleuthing out errors, and other useful programming tools. The most popular IDE for Windows is Visual Studio, while Apple offers the Xcode IDE. Most IDEs can support several different programming languages. The IDE will also *compile* a programmer's code, turning the higher-level lines of instructions that are readable by humans into lower-level instructions expressed as the patterns of ones and zeros that are readable by a computer's microprocessor.

Look at the side of a box of commercial software and you're likely to see system requirements that specify the operating system and processor that the software is designed for (e.g. 'this software works on computers with Windows 7 and Intel-compatible processors'). Wouldn't it be great if software could be written once and run everywhere? That's the idea behind Java – a programming language developed by Sun Microsystems.

Java programmers don't write code with specific operating system commands (say for Windows, Mac OS X, or Linux), instead they use special Java commands to manage the display and other hardware. Java programs can run on any computer that has a Java Virtual Machine, a software layer that interprets Java code so that it can be understood by the operating system and processor of a given computer. Java's platform independence is its biggest selling point. Many web pages execute Java applets to run the animation you might see in advertisements or games. Java has also been deployed on over 6 billion mobile phones worldwide, and is popular among enterprise programmers who want to be sure their programs can scale from smaller hardware up to high-end supercomputers. However, Java has not been popular for desktop applications. Since Java isn't optimized to take advantage of interface elements specific to the Mac or Windows, most Java desktop applications look clunky and unnatural. Java code that runs through the JVM interpreter is also slower than code compiled for the native OS and processor that make up a platform.

Scripting languages are the final category of programming tool that we'll cover. Scripting languages typically execute within an application. Microsoft offers a scripting language called VB Script (a derivative of Visual Basic) to automate functions in Office. And most browsers and web servers support JavaScript, a language that helps make the web more interactive (despite its name, JavaScript is unrelated to Java). Scripting languages are *interpreted* within their applications, rather than compiled to run directly by a microprocessor. This makes them slower than the kinds of development efforts found in most commercial software. But most scripting languages are usually easy to use, and are often used both by professional programmers and power users.

KEY TAKEAWAYS

- Programs are often written in a tool called an IDE, an application that includes an editor (a sort of programmer's word processor), debugger, and compiler, among other tools.
- Compiling takes code from the high-level language that humans can understand and converts them into the sets of ones and zeros in patterns representing instructions that microprocessors understand.
- Popular programming languages include C++, C#, Visual Basic, and Java.
- Most software is written for a platform – a combination of an operating system and microprocessor.
- Java is designed to be platform independent. Computers running Java have a separate layer called a Java Virtual Machine that translates (interprets) Java code so that it can be executed on an operating system / processor combination.
- Java is popular on mobile phones, enterprise computing, and to make web pages more interactive. Java has never been a successful replacement for desktop applications because user interface differences among the various operating systems are too great to be easily standardized.
- Scripting languages are interpreted languages, such as VB Script or Java Script. Many scripting languages execute within an application (like the Office programs, a web browser, or to support the functions of a web server). They are usually easier to program, but are less powerful and execute more slowly than compiled languages.

TOTAL COST OF OWNERSHIP (TCO): TECH COSTS GO WAY BEYOND THE PRICE TAG

Managers should recognize that there are a whole host of costs that are associated with creating and supporting an organization's information systems. Of course, there are programming costs for custom software; and purchase, configuration, and licensing costs for packaged software, but there's much, much more. There are costs associated with design and documentation (both for programmers and for users). And any new programs should be tested thoroughly across the various types of hardware the firm uses, and in conjunction with existing software and systems, *before* being deployed throughout the organization. Any errors that aren't caught can slow down a business or lead to costly mistakes that could ripple throughout an organization and its partners. Studies have shown that errors not caught before deployment could be 100 times more costly to correct than if they were detected and corrected beforehand[13]. Consider the fate of FoxMeyer and Hershey Foods.

Once a system is 'turned on', the work doesn't end there. Firms need to constantly engage in a host of activities to support the system:

- providing training and end user support;
- collecting and relaying comments for system improvements;

[13] Charette, 2005.

- auditing systems to ensure *compliance*, i.e. that the system operates within the firm's legal constraints and industry obligations;
- providing regular backup of critical data; planning for redundancy and disaster recovery in case of an outage; and
- vigilantly managing the moving target of computer security issues.

With so much to do, it's no wonder that firms spend 70 to 80% of their IS budgets just to keep their systems running[14]. The price tag and complexity of these tasks can push some managers to think of technology as being a cost sink rather than a strategic resource. These tasks are often collectively referred to as the *total cost of ownership*, or *TCO*, of an information system. Understanding TCO is critical when making technology investment decisions.

Why Do Technology Projects Fail?

Even though information systems represent the largest portion of capital spending at most firms, an astonishing one in three technology development projects fail to be successfully deployed[15]. Imagine if a firm lost its investment in one out of every three land purchases, or when building one in three factories. These statistics are dismal! Writing in *IEEE Spectrum*, risk consultant Robert Charette provides a sobering assessment of the cost of software failures, stating "the yearly tab for failed and troubled software conservatively runs somewhere from $60 billion to $70 billion in the United States alone. For that money, you could launch the space shuttle 100 times, build and deploy the entire 24-satellite Global Positioning System, and develop the Boeing 777 from scratch—and still have a few billion left over."

Why such a bad track record? Sometimes technology itself is to blame, other times it's a failure to test systems adequately, and sometimes it's a breakdown of process and procedures used to set specifications and manage projects. A multi-million dollar loss on the NASA Mars observer was traced back to Lockheed Martin contractors using English measurements, while the folks at NASA used the metric system[16]. The agency and the contractor had to deal with the unwanted attention of losing a $125 million taxpayer investment because a bunch of rocket scientists failed to pay attention to third grade math. When it comes to the success or failure of technical projects, the devil really is in the details.

Projects rarely fail for just one reason. Project post-mortems often point to a combination of technical, project management, and business decision blunders. The most common factors include[17]:

- Unrealistic or unclear project goals
- Poor project leadership and weak executive commitment
- Inaccurate estimates of needed resources
- Use of immature technology
- Unmanaged risks
- Inability to handle the project's complexity
- Sloppy development and testing practices

---

[14] Rettig, 2007
[15] Dignan, 2007
[16] Lloyd, 1999
[17] List largely based on Charette, 2005

- Badly defined system requirements and allowing 'feature creep' during development
- Poor reporting of the project's status
- Poor communication among customers, developers, and users
- Poor project management
- Stakeholder politics
- Commercial pressures (e.g. leaving inadequate time or encouraging corner-cutting)

Managers need to understand the complexity involved in their technology investments, and that achieving success rarely lies with the strength of the technology alone.

But there is hope. Information Systems organizations can work to implement procedures to improve the overall quality of their development practices. Mechanisms for quality improvement include the Capability Maturity Model (CMM), which gauges an organization's process maturity and capability in areas critical to developing and deploying technology projects. The five level criteria helps an organization assess its capabilities and provides guidelines for improving overall processes and procedures.

Firms are also well served to leverage established project planning and software development methodologies that outline critical businesses processes and stages when executing large-scale software development projects. The idea behind these methodologies is straightforward – why reinvent the wheel when there is an opportunity to learn from and follow blueprints used by those who have executed successful efforts. When methodologies are applied to projects that are framed with clear business goals and business metrics, and that engage committed executive leadership, success rates can improve dramatically[18].

KEY TAKEAWAYS:
- The care and feeding of information systems can be complex and expensive. The total cost of ownership of systems can include software development and documentation, or the purchase price and ongoing license and support fees, plus configuration, testing, deployment, maintenance, support, training, compliance auditing, security, backup, and provisions for disaster recovery. These costs are collectively referred to as TCO, or a system's total cost of ownership.
- Information systems development projects fail at a startlingly high rate. Failure reasons can stem from any combination of technical, process, and managerial decisions.
- IS organizations can leverage software development methodologies to improve their systems development procedures, and firms can strive to improve the overall level of procedures used in the organization through models like CMM. However, it's also critical to engage committed executive leadership in projects, and to frame projects using business metrics and outcomes to improve the chance of success.

[18] Shenhar and Dvir, 2007

About the Author:

John Gallaugher is a member of the Dept. of Information Systems in Boston College's Carroll School of Management. Prof. Gallaugher teaches courses and conducts research at the intersection of technology and strategy. He leads the School's TechTrek programs, co-leads the Asian field study program, and has consulted to and taught executive seminars for several organizations including Accenture, Alcoa, Brattle Group, ING Group, Partners Healthcare, Patni Computer Systems, Staples, State Street, and the U.S. Information Agency. Writings, podcasts, course wiki, blog (The Week in Geek), and research by Prof. Gallaugher can be found online at www.gallaugher.com.

This reading is available to faculty for non-commercial use. Enjoy! If you do use it, please send an e-mail to john.gallaugher@bc.edu. More chapters and cases will follow in Professor Gallaugher's forthcoming book "Information Systems: A Manager's Guide to Harnessing Technology", to published (in both free online and low-cost print version) by Flat World Knowledge (FlatWorldKnowledge.com). Thanks!

REFERENCES

Arrington, M., “Rearden Commerce: Time For The Adults To Come In And Clean House”, *TechCrunch*, April 5, 2007.

Arrington, M., “2008: Rearden Commerce Has A Heck Of A Year”, *TechCrunch*, Jan. 13, 2009.

Charette, R., “Why Software Fails”, *IEEE Spectrum*, Sept. 2005.

Dignan, L., “Survey: One in 3 IT projects fail; Management OK with it”, *ZDNet*, Dec. 11, 2007.

Drummond, M, “The End Of Software As We Know It”, *Fortune*, Nov. 19, 2001.

Edmondson, G., “Silicon Valley on the Rhine”, *BusinessWeek International*, Nov. 3, 1997.

Fortt, J., “Tech Execs Get Sexy”, *Fortune*, Feb. 12, 2009.

Hamm, S., “Making Computers Based on the Human Brain”, BusinessWeek, Nov. 20, 2008.

Kerstetter, J., “The Web At Your Service”, *BusinessWeek*, March 18, 2002.

Koch, C., “Nike Rebounds: How (and Why) Nike Recovered from Its Supply Chain Disaster”, *CIO*, June 15, 2004.

Krill, P., “SOA Gets an Obituary”, *InfoWorld*, Jan. 5, 2009.

Lloyd, R., “Metric mishap caused loss of NASA orbiter”, *CNN*, Sept. 20, 1999

Lyons, D., “Cheapware”, Forbes, Sept. 6, 2004.

McCarthy, J., “The Standards Body Politic”, *InfoWorld*, May 17, 2002

McMillan, R., “Gone in Two Minutes”, *InfoWorld*, March 27, 2008.

National Petroleum News, “Petroleum industry continues to explore EDI”. National Petroleum, Vol. 90, Issue 12, Nov. 1998

Richardson, B., “Rearden Commerce: Software for Road Warriors”, AMR Research, June 6, 2008.

Robertson, J., “IBM sees better-than-expected 2009 profit, earns US$4.4 billion in Q4”, Associated Press, Jan. 20, 2009.

Robinson, A., and Dilts, D., “OR & ERP”, *ORMS Today*, June 1999.

Schonfeld, E., “At Rearden Commerce, Addiction is Job One”, TechCrunch, May 6, 2008.

Shenhar, A., and Dvir, D., *Reinventing Project Management: The Diamond Approach to Successful Growth and Innovation*, Harvard Business School Press, August 2007, Boston, MA

W3C, “Web Services Architecture”, W3C Working Group Note, Feb. 11, 2004.