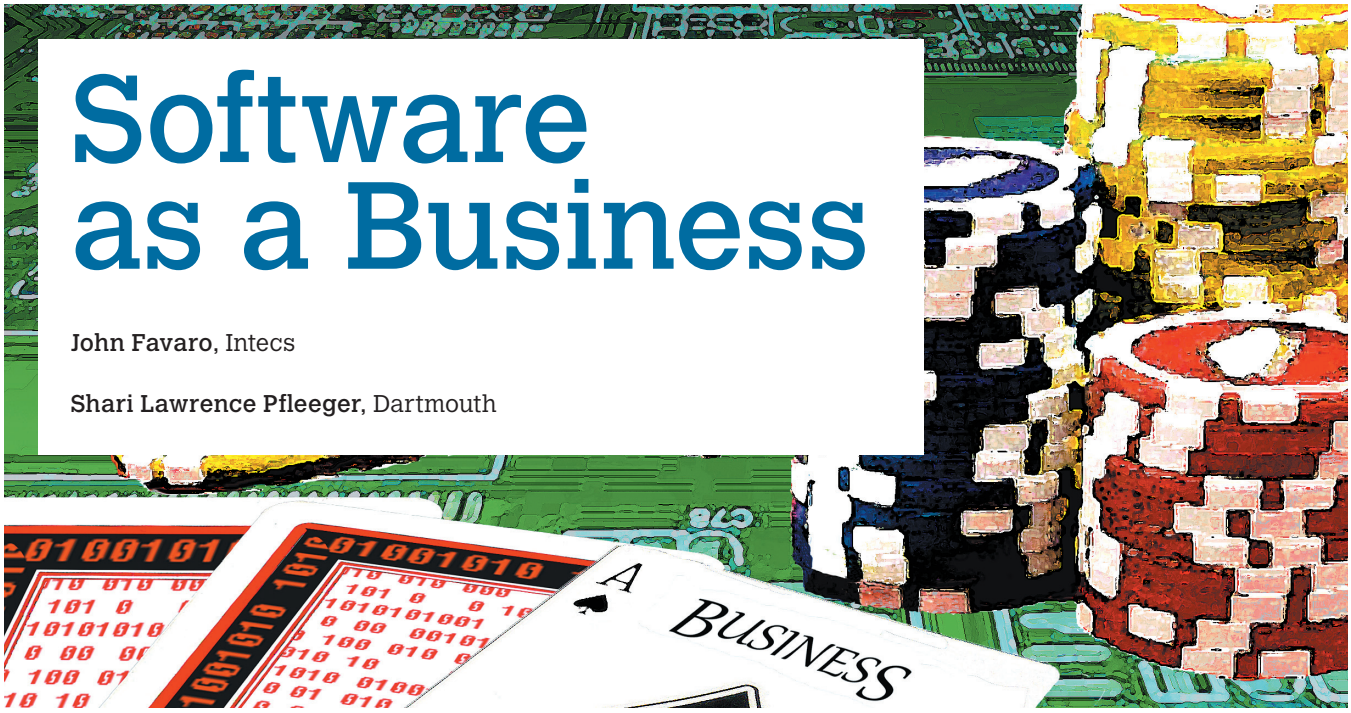
Software as a Business

John Favaro, Intecs

Shari Lawrence Pfleeger, Dartmouth



IN THE DECADES since World War II, software has played an increasingly important role in most aspects of business. Indeed, it's difficult today to find a business function that doesn't involve computers and software in some way. Recent reports have documented the enormous share of the overall economy that software now occupies—for example, the US Bureau of Economic Analysis reports that computer equipment and software contributed 11.6 percent to the US gross domestic product in the last quarter of 2010 (www.bea.gov/newsreleases/national/gdp/gdpnewsrelease.htm), and the European Commission recently financed a half-million Euro study on the "Economic and Social Impact of Software and Software Based Services," the first step in defining a European software strategy (http://cordis.europa.eu/fp7/ict/ssai/docs/study-sw-report-final.pdf).

In the beginning, businesses considered software as a way to automate processes, contributing to productivity by speeding up what was already being done. But over time, software became recognized not just as an automation tool but more broadly as a strategy for providing products and services not yet offered. Thus, software developers have broadened their perspective, creating architectures and support strategies that fit the business model in which software products and services will be embedded.

Development organizations no longer describe service-oriented architectures solely in terms of their technical characteristics; instead, they view them in relationship to software as a service (SaaS) business models. Increasingly, they mention component-based software development in the same breath as software ecosystems and frame agile development processes in terms of "delivering business value." The savvy software developer rarely mentions the cloud apart from a discussion of its implications for new ways of doing business in software. In short, today's software engineering community is focusing on the software business.

Viewing Engineering from a Business Perspective

At least three key questions confront the software industry today:

- From a corporate perspective, how do software creation and support organizations address an enterprise's existing business model? That is, how does a successful enterprise embrace software products and services to preserve or improve its competitiveness?
- How does an enterprise whose primary focus is software (components, applications, and services) find a successful business model?
- From a software engineering perspective, what architectures and support strategies yield products and services that enhance an enterprise's business model?

0740-7459/11/$26.00 © 2011 IEEE

The answers to these questions inform any analysis of how, when, why, and whether a business should incorporate software into its products, processes, and services. For example, a recent article reports that "Some … Japanese companies risk becoming too obsessed with sophisticated engineering, to the point that they overlook points that appeal to consumers, like ease of use and design."[1]

Similarly, the mobile devices industry has given rise to a business model for small, low-cost applications (apps) that can be sold in large quantities—creating an unexpected competitor to the open source model whereby software is "almost free." Consequently, advocates of open source business models are rethinking the value of a closed market, given the success of Apple's iPod and iPad.[2]

Choosing the Right Model

So how does an enterprise know which model to pick from the many that have arisen in the last decade? And how do software engineers map the model to the architecture? The first step is to determine whether the software should be marketed as a product or a service. The latter notion is relatively new. In his 2004 book, Michael Cusumano discusses this pivotal decision, describing SaaS as a business strategy, not just an accidental outcome.[3]

The nature of the software influences the business, and vice versa. For example, as software becomes ubiquitous, a business model for embedded software is emerging with its own particular characteristics. Because it's marketed in the context of a larger system, the software's business model and its "host" are necessarily intertwined. For instance, the AUTOSAR initiative has evolved as an approach and architecture that encourage the development of a market for embedded components in automotive systems.

Similarly, a business model based on outsourcing has increased in the past decade, especially in the form of *offshoring*. Here, the software business model's changes are similar to those of other businesses: the business evolves as labor skills, location, and costs change. A variant model is arising around the idea of *crowdsourcing*, where a pool of talent is tapped for a short time to do software development, with no fixed long-term contractual relationship.

How does an enterprise compare and contrast business models? One way is to itemize the ways that it can produce revenue and then examine the risks:

- Which products and services appeal to clients?
- Which clients are willing to pay for licenses, maintenance, updates, customization, and other services?
- Which models are most likely to feel the impact of national or global economic shifts?
- Which models can scale up or down to meet different clients' needs?
- How much skill is needed to sell and support the products and services; how often will that skill required updating; and at what cost?

The many considerations involved in business model choice are also affected by external forces such as standards and regulation.

Whither Standards?

In the past, standards and standard interfaces have emerged on the basis of well-known technical issues in the software industry. But now that business models play a role, a standard's business implications aren't just important—they can actually influence the standard's evolution. For instance, just as many automobile parts evolved toward a standard to make construction, training, and maintenance (and driving!) more uniform across brands, so, too, might software architectures become more standardized, enabling one application to be seamlessly loaded or swapped with another.

As software becomes ubiquitous, a business model for embedded software is emerging with its own particular characteristics.

As with other kinds of standards, governments are intervening to establish standards that encourage competition and industry growth, even creating and shaping the markets themselves. For instance, the European Commission has established a "Framework for Pan-European eGovernment Services." But the framework, which supports open source over other business models, is opposed by the Business Software Alliance; the BSA prefers technology neutrality that also permits competition among commercial software businesses (www.bsa.org/country/Public%20Policy/innovation/tech-neutrality.aspx). Tensions between the goals of standards and the goals of those who must use them will continue as software business evolves.

ABOUT THE AUTHORS

JOHN FAVARO is a senior consultant at Intecs, where he is deputy director of research. His current interests include value-based management of IT and strategic positioning of software enterprises. Favaro has an MS in computer science from the University of California, Berkeley. Contact him at john@favaro.net.

SHARI LAWRENCE PFLEEGER is director of research for Dartmouth University's Institute for Information Infrastructure Protection, a consortium of 27 major universities, national laboratories, and nonprofit research centers. Her research interests include cybersecurity, technology transfer, and risk management. Pfleeger has a PhD in information technology and engineering from George Mason University. Contact her at pfleeger@dartmouth.edu.

Ancillary Effects

The software business has secondary effects, involving consultancies that assist enterprises in finding and enacting successful business models. For example, "innovation management" and "business analysis" have emerged as new specialties that augment or even supplant traditional activities such as requirements engineering. One innovation management company offers software to manage crowdsourcing contests for new ideas, to encourage employee innovation within an enterprise, and to predict events and outcomes. Another's software provides an innovation engine to oversee the review process and a platform that manages intellectual property rights (IPR).

In fact, IPR looms larger as software provides more essential product functionality. For example, Kevin Fu's analysis of medical devices reveals that manufacturers are increasingly using software to implement essential functionality. Indeed, a milestone occurred in 2006, after which more than half the medical devices on the market now rely on software in some way.[4] Other products are increasing their software reliance, too, with some cars having more software than some airplanes:

> *The Chevrolet Volt plug-in hybrid uses about 10 million lines of computer code to shunt power seamlessly among the car's battery pack, power inverter, drive motor, gas engine, generator and other subsystems. By comparison, Boeing's new 787 Dreamliner relies on a mere eight million lines of code. Automakers therefore view leadership in control software as strategically vital, said Eric Fedewa, head of powertrain forecasting at IHS Automotive, a consulting firm based in Englewood, Colo.*[5]

The European Patent Office (EPO) is wrestling with the issue now because it's being pressed to make software patents legal. Even Philips, the Dutch electronics giant, has submitted software patent applications to the EPO—a big move for a country that abolished patents in the early 20th century. Software as a business is clearly influencing many a corporate bottom line.

In This Issue

For this special issue, we sought software business articles that are relevant to developers and managers. The four contributions highlight important aspects of software business that influence not only requirements elicitation and design but also service provision and long-term system support.

In "Software Company Business Models," Karl Michael Popp provides an overview of the main categories of business models and describes a typology derived from successful software companies. He notes that a business model describes the company's goods and services and related revenue model, but not how the business is managed. Using three actual software companies as examples, he shows how the right business model can create a competitive advantage.

Figuring out how and what users will pay for a company's software is an essential element of a business model. In "Matching Open Source Software Licenses with Corresponding Business Models," Juho Lindman, Matti Rossi, and Anna Paajanen describe open source software licenses. Because the license circumscribes what a user can and can't do with open source code, developers must pay careful attention to the rules for licensed components. In addition to describing common licenses and their characteristics, the authors discuss variables that influence licensing decisions, such as developer motivation, externalities, and company size and control.

In "Sharing Source Code with Clients: A Hybrid Business and Development Model," Mikko Riepula considers licenses that are open- and closed-source hybrids. A *client-shared source* model, in which the company shares proprietary code with select clients for further development, is an example of a licensing model innovation that's as important as technological innovation in advancing the software business. It's partly based on the keen observation that there's often more business value locked up in client relationships than in a product's source code.

Talk of business model innovation

is fine, but the models must be tested with experience. In "Developing Cloud Business Models: A Case Study on Cloud Gaming," Arto Ojala and Pasi Tyrväinen track the changing business model of a small gaming company over a 10-year period, as it adjusts to new challenges (such as shifting technologies) and opportunities (such as a virtual store's ability to support "long tails" of low-volume products).

Is the software business really "different" from other businesses? Some observers believe that software's special characteristics, such as malleability, negligible duplication costs, and low distribution costs, make it unique. Others insist that it's only a matter of degree, not of kind. Did Windows achieve a dominant market share in the operating system market through a "network effect" that's peculiar to the software industry, or can we explain its dominance through normal business analyses—such as building high barriers to entry or using skillful marketing techniques? Can we attribute the success of Apple's mobile products to some kind of "disruptive technology" or the rise of a new "ecosystem" around social networking, or is it part of a normal substitution effect that occurs in all evolving business sectors? The debate will no doubt continue. But as it does, the software industry will continue to transform itself. For this reason, our special issue on software business provides thoughtful analysis of the key issues that will shape not only the business of software but the roles and responsibilities of the developers who enable it.

References

1. M. Fackler, "In Toyota Mess, an Economic Lesson for Japan," *New York Times*, Feb. 8, 2010; www.nytimes.com/2010/02/09/business/global/09toyota.html.
2. S. Johnson, "Everybody's Business: Rethinking a Gospel of the Web," *New York Times*, 11 Apr. 2010; www.nytimes.com/2010/04/11/technology/internet/11every.html.
3. M. Cusumano, *The Business of Software*, Free Press, 2004.
4. Institute of Medicine, "Public Health Effectiveness of the FDA 510(k) Clearance Process: Measuring Postmarket Performance and Other Select Topics—Workshop Report," Workshop Report, Nov. 2010.
5. Lindsay Brooke, "Computer Code an Increasingly Precious EV Commodity," *New York Times*, 23 Jan. 2011; www.nytimes.com/2011/01/23/automobiles/23SPIES.html.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.