A Market-based Perspective on Information Systems Development

Steve Sawyer

The ongoing shift in how software is made—from user organizations developing their own to a market in which vendors package ready-to-install products—means fundamental changes in how information systems are developed.

Information systems development (ISD) is best understood as a market phenomenon, a perspective highlighting how software is developed and who performs that development and sells the related products and how they are introduced to users. Here I emphasize the increasing specialization of software producers (developers and vendors) as distinct from software-consuming organizations. I also contrast software product development with ISD, emphasizing my view of the worldwide software product market, exploring important implications for consumers.

Worldwide software sales rose 280% from 1986 to 1995 [2] and are expected to double again by 2002, fueling market growth, along with the market capitalizations of Microsoft, Oracle, SAP, and other major vendors. Software comes as either packaged (commercial, or shrink-wrap) or made-to-order (custom, or one-off). I distinguish between software producers (vendors of packaged software and software houses) developing, manufacturing, and distributing software and software-consuming organizations acquiring (buying) and using it. Software producers include such huge organizations as EDS, IBM, Lockheed-Martin, Microsoft, Oracle, SAP, as well as thousands of smaller firms. While a software consumer can be an individual or an organization, I focus on organization-level consumption. Thus, when Microsoft buys a license for, say, SAP R/3 products for managing financial operations and product inventories, it becomes a software consumer.

Differentiating between producer and consumer points up that the boundaries between these roles are increasingly organizational in nature. An underlying assumption is that the changes in development are less dramatic than the changes in acquiring and installing software-based information systems [3, 4, 6]. I thus view these issues from the perspective of a software consumer to reveal how the software product market is changing ISD.

Software Products and Information Systems

Software products are discrete components increasingly sold as packages and as ready-to-attach modules. An information system includes the software, hardware, people, and rules that make a collection of software products work for the consumer. Software is often the central element, though a system almost always includes a number of software products.

Distinguishing software products from an information system highlights the number of components

that have to be assembled to create a working system. That number reflects the evolution of software-based systems from the approach typical in the early 1990s of being designed in-house and built from the code up by each user organization's own IT staff to today, when many (if not most) of a system's pieces are bought ready to be installed. As software development, manufacturing, and distribution increasingly become the work of specialized organizations, software-consuming organizations increasingly assemble pieces, not build them. That is, software consumers focus on ISD, while vendors focus on developing packaged products.

Table 1. Traditional SDLC and market-oriented SDLC compared.

Traditional SDLC	The two parts of a market-based model of ISD	
	Consumer SDLC	**Producer SDLC**
Planning Project selection Project initiation	System planning System selection System initiation	Product planning Product selection Product initiation
Requirements analysis	System needs analysis Product function analysis Gap-fit comparison	Product requirements defined
Design Programming Test and install	 System installation	Product design Product development Product test and ship
Maintain	System support and upgrade	Product support

The software product market represents a forum for exchanging goods and services between producers and consumers. A market is a basic economic form arising when there is a need for a product or service and consumers have difficulty producing that good or service for themselves [11]. The consumer's need is an economic incentive for a producer to accept the risks inherent in creating something to sell.

A market can be defined in several ways, though certain principles or characteristics are common to all definitions. For example, markets are considered "visible"; that is, producers and consumers can find them. Observing any technology-oriented media outlet substantiates the software product market's visibility. Markets are also characterized by competition and organizational specialization. That is, due to competitive pressure, organizations involved in software development and distribution focus their efforts on a limited number of goods and/or services in order to compete effectively. For example, the giant German software manufacturer SAP AG makes and sells enterprise-level application software, not operating systems or databases. Still other companies, such as AMS and Anderson, provide software services. That is, they help install and integrate software (and hardware) products into consumer organizations' systems. The ongoing discussion as to whether Microsoft's role in the PC software products market is open (despite being found guilty in federal court of monopolistic practices last year) assumes a market perspective relative to the flow of information.

Another way to describe a market is that it forms when information "asymmetries," or gaps between information need and delivery, exist between producers and consumers. An information asymmetry means it is difficult for one party to gain knowledge held by another party to make that good or perform that service. A market provides a means to substitute product or service transfer for knowledge transfer [11]. For example, some of the core information asymmetries in ISD, including requirements analysis and implementation, first identified by Gerry Weinberg [10] and Fred Brooks [1] more than 25 years ago, have not been alleviated by the evolutionary (albeit valuable) developments in software engineering. Despite being difficult for developers to build and deliver the kinds of systems users want, it is even more difficult for users to build and install their own systems.

Conversely, if knowledge transfer is the goal of the exchange between buyers and sellers, a hierarchy works best [11]. This means of delivering ISD knowledge is one reason traditional software process methods, such as the Software Engineering Institute's Capability Maturity Model, advocate for an internal software process/methods group to facilitate (internal) knowledge transfer. Buying a piece of software does not improve a consumer organization's ability to make that software. Moreover, buying a piece of software does not mean the consumer now has an information system.

Market-Oriented Versus Traditional ISD

To illustrate how a software product market changes ISD, I contrast a market-oriented approach with a simplification of the traditional "waterfall" model approach [9], or a stage model outlining the steps in software development. Despite its well-known shortfalls, the waterfall model encompasses many of the functions characterizing the ISD approach and is often honored in spirit more than in detail. I could also contrast the market-based approach to other ISD models, including rapid application development, evolutionary/spiral, and object-ori-

Table 2. Changing analysis, implementation and support for consumer organization SDLC.

Traditional SDLC	Consumer SDLC
Requirements analysis: Develop detailed requirements Map requirements to specifications Create specification document	System needs analysis: High-level functional needs (for RFP) Critical features/functions needed Product function analysis: Market identification Product features/functions assessment Product comparison/selection Gap-fit comparison: Between product and organization. Initial identification of customizations
Test and Install: Assemble modules Test for reliable operations Test for functional specifications Install/distribute software Train users on new system	System installation Detailed requirements analysis: Develop data structures and business rules Decide configuration table settings Additional customization Business process changes To fit new functionality To accommodate missing functionality Install/distribute software: Train users: On new system In new processes

ented. However, my goal here is to explain the market-based perspective and highlight how the emerging software product market is reshaping ISD.

The waterfall model is both more broadly known and just as useful as any of the other approaches [3, 6]. Implicit are at least two relevant assumptions: that ISD takes place within one organization (or, at least, is controlled by that organization when hiring contractors and consultants to build a custom product), thus reflecting vertical integration (a hierarchy); and that it is focused on building, not buying, software. These assumptions are appropriate because building (not buying) software was the waterfall model's intended purpose. However, these assumptions also obscure the market-oriented forces arising when software is bought, not built.

Market-based model. Table 1 outlines the emerging market-based model of ISD, which focuses on the stages of ISD performed by the consumer and the goals and the participants and their degrees of involvement at each stage. The phases of ISD now performed by software consumers include system planning, selection, initiation, product-feature analysis, installation, and support. Table 2 outlines the differences between traditional ISD and the market-oriented perspective's consumer ISD regarding analysis and installation.

For consumers, system planning, selection, and initiation are similar to the traditional waterfall phases, with two significant differences. First, a commitment to purchase means a significant investment, thus attracting senior management's attention and involvement. That is, the costs of buying large systems means acquiring software products is both a highly visible organizational activity and a significant capital investment. This scale of investment and strategic benefit implies senior management, not only IT leaders, is typically involved in software product selection. Moreover, as the cost of IT product investment increases, decisions are increasingly made by senior, not only IT, managers.

Second, and partly because of the combination of visibility and cost, these early phases are often performed in conjunction with third parties, including enterprise resource planning (ERP) vendor representatives and strategic IT consultants. Consumer senior management's goal is developing a plan linking perceived organizational needs to known and emerging software products. The focus is senior management involvement, both line and technical; end users are not especially involved. The result is a request for proposal (RFP) or request for information (RFI) outlining how the combined market/needs analysis and implementation steps should take place. Selection is often driven by expectations of the technical trajectories of potential future vendor products as much as by specific internal needs. That is, a consumer organization has difficulty deviating from where, say, SAP R/3's designers and developers decide to take the product in the interests of their own business performance and perceived market needs.

The system-needs-and-product-features-analysis phase of market-based ISD grows out of RFP/RFI development, often supported by vendor representatives and IT consultants. It consists of two components: One is a high-level analysis of organizational needs whose goal is identification of critical organizational needs. The other involves identification and comparison of the functions and features of the various potential product solutions, including a "gap-fit" analysis, or the matching of product features to organizational needs. This analysis helps identify both the technical adjustments, or customizations, needed and the organizational processes that must be changed when the product's functions do not support these

processes. Typically, however, only high-level requirements are used by analysts to help assess which packaged or custom-built software product represents the best fit with the consumer's business criteria. The transition from analysis to system installation begins with the letting of contracts among the consumer, product vendor, and consultant to support implementation.

System installation means installing a vendor's product(s), generally with the help of third-party consultants contracted to customize the code and establish the proper table, parameter, and option settings, and, perhaps, guide the installation process as well. Typically, other third parties provide training and help-desk support. The goal of installation is to get the new software operating in a way that allows individual end users to help their organizations improve overall business performance. The installation phase is also where the contract drawn up at the conclusion of the features-versus-needs analysis process becomes especially important because it guides implementation, including who does what, as well as how costs are assigned to these activities.

ISD sequencing, along with the people involved and software functionality and its integration into business processes, means installation is a project of guided discovery, since identifying risk is difficult in advance. For example, it is common for most organizations installing an ERP system to have dozens of user groups, project teams, and ad-hoc committees involving hundreds, even thousands, of line workers, supporting technologists, and consultants working on this aspect of the project. Project planning must be adaptable, as system installation reflects the mutual adaptation of new software-based processes to existing organizational processes. Moreover, the need for flexibility suggests increased productivity lags installation, as users adapt to new software-based functionality and the altered business processes demanded or permitted by the new software.

Installation involves many detailed analysis issues. In the traditional systems development life cycle (SDLC), they were handled earlier, as part of the requirements analysis stage. In the market model of SDLC, detailed analysis comes after purchase. It is only during installation that users become deeply involved for the first time in assessing how the software meets their needs.

System support means maintaining the system's current, supported, version(s) of the software product(s). This phase of consumer SDLC represents the management of a diverse set of third-party service providers, as well as software and hardware vendors. Still, despite major changes in how consumer organization IT staff maintains a system's various computing components, end users see and experience a coherent system. Being responsible for supporting the installed system has critical implications for consumers. The range of vendors and consultants demands that consumers maintain complex multi-party service contracts.

Implications

The software services market will be increasingly important for the foreseeable future. Consumer organizations will learn to view software producers (vendors) as partner *and* producer while reconsidering some assumptions about ISD.

Software services. The software product market exists in part because of the participation of other non-software-vendor participants, including consulting firms, system integrators, and third-party support, training, and service organizations, as well as other software producers, whose products bolt onto and extend the functionality of larger products. These additional players help bring software consumers and producers together, adding value to their transactions. That is, the software product market also gives rise to a software services market. For example, my data indicates service costs for an ERP installation are typically three times the combined costs of the related software and hardware alone (see the sidebar "Data Sources").

Consumer's view. The typical software product vendor's SDLC process is less important to the consumer than is the system's purpose. That is, vendors now focus on developing products, not systems [3, 5, 6, 12]. Thus, in the early stages of a producer's SDLC, the producer focuses on adapting its products to better reflect market needs, not the needs of a particular user organization. This market perspective suggests the consumer organization views the software producer's increased product attention as leading to changes in the way it interacts with vendors, reduced interest in (and visibility for) the producers software engineering efforts, and developers' minimal involvement in the consumer organization's implementation effort [2].

The increased separation of users from developers by both organizational and market boundaries changes how these groups interact. In traditional SDLC, early and close links between users and developers was considered critical. Today, software consumers and producers use a variety of intermediated means to communicate their needs to developers. For example, packaged software developers build to requirements gleaned from a variety of sources, including help-desk call-log analysis, market research, product reviews, and user groups, of which direct cus-

tomer contact is one of the least likely means [7]. Consumers may communicate requirements via help desks, user groups, and contract documents, including RFPs and RFIs.

Another issue complicating the vendor-user relationship is that much of what we traditionally view as software engineering and development is now opaque to most consumer organizations. Since production is separated from consumption, software engineering methods, techniques, and tools are less important to the consumer than is the outcome of their use. That is, vendors are being evaluated by their potential customers on the basis of their products, not their processes. This product focus permeates how vendors develop software and is a fundamental aspect of why packaged software development differs from traditional in-house development [4].

A product focus also underlies how software maintenance changes in the software products market. For example, software vendors separate corrective maintenance, including patches, from other forms of software maintenance. Patches and workarounds are often provided to licensees at no cost beyond a subscription fee. However, the changes needed to smooth out poorly done but operable software functions become the basis of new releases for which vendors charge additional, often highly profitable licensing fees. That is, most of what was once maintenance in traditional SDLC now forms the basis of a product's next release and thus serves to generate additional revenue for the vendor over a number of years.

Software vendors often minimize their role in implementation. A dominant goal of software vendors is to "ship" their products, leaving it to others to implement them in consumer organizations [4]. This is but one example of firm-level specialization in a market; builders build and integrators integrate. It also means opportunities for other firms to focus on implementation and become market intermediaries between buyers and sellers. In fact, without intermediaries, the software market would fail, since transactions between vendors and consumers are often quite complex.

Perhaps one of the most visible differences between a market-oriented ISD model and traditional SDLC is the increased importance of consultants, system integrators, and support firms that help bridge the consumer-producer relationship. For example, IT consultants are typically involved in helping consumer organizations make strategic IT investments. Moreover, IT consultants are often hired to assist a consumer organization with installation of software and incorporation of that software into its existing information systems. Other third parties may provide training, technical support, and even additional (add-on) products to extend the functionality of the system.

These changes to ISD suggest there is a need to develop consumer-focused techniques, such as work-process analysis, gap-fit analysis, and market analysis. New work-process analysis techniques, including scenarios that help model typical usage patterns assist consumers assessing how well purchased software would actually meet their needs. More robust gap-fit techniques are also needed to enable consumers to assess the differences between software product functionality and their current organizational processes and needs. While these techniques exist in other areas of product-purchase evaluation, IT analysts in software-consuming organizations also need to adapt and incorporate more sophisticated skills in market analysis.

The market-oriented perspective on ISD also means that, for consumer organizations, current thinking regarding their IT people skill mix, roles, and governance structures may need to be reassessed to better reflect the tasks these people are now being asked to perform. For example, IT people in consumer organizations should know about contracts and contractual negotiations, the products and services available in the market, and the values and costs of third-party support. IT people also need to learn the tactics involved in influencing vendors, including user groups, focus groups, and product testing. Moreover, their role within consumer organizations increasingly involves supporting standards, providing technical and market trend analysis, and supervising third-party work to install, maintain, and support a system's various components.

Changes in the roles of and skills needed by the consumer organization's IT professionals suggest that approaches to teaching ISD should evolve to better

Data Sources

In addition to drawing from the academic, professional, and trade literature on ISD and software development, I have drawn from two sets of my own empirical data derived from interviews, observations, and surveys. One I collected from nearly 100 software development teams at 24 companies making custom and packaged software, including operating systems, languages, and an array of applications. The other derive from two in-depth multiyear case studies I conducted 1995—2000. Both followed implementation of ERP software into a 3,000-employee $500 million annual revenue U.S. services organization and a 600-employee $500 million annual revenue U.S. manufacturing organization. C

reflect the increasing importance of specialization implied by a market perspective. The way ISD is taught to IT professionals should include the new forms of analysis, roles in software development (and how they differ in ISD and implementation), and the meanings of use and maintenance. This new way of teaching ISD to IT professionals may lead to greater differences in the way computer science and information systems students are taught. The former is likely to focus more on construction of requirements plans and development of software; the latter is likely to focus more on support, implementation, market and needs analysis, and the contractual issues surrounding ISD.

Market Forces

The software product market will evolve rapidly in the near future, as other potentially market-shaping forces emerge. For example, Web-based development focuses on content, submerges the developer behind that content, and promotes rapid system changes in response to user needs. This approach derives, in part, from the increasing standardization of interfaces between both computers and other devices and between users and computers. Standardization also highlights the rise of component-based software in Web-oriented ISD with the potential for developing and deploying over the Web plug-in software modules designed to fit a set of base products.

The emerging roles of standard software components, including application protocol interfaces and protocols, is another potential force. Standards help reduce the variability of base components and architectures, allowing increased component use and reuse. These standards are often market-based in that they are shaped by the practices of several influential vendors or consumers. Such control gives the vendor (or consumer) the ability to influence its own role in the software product market [8].

The very existence of such a large and growing market makes regulation another likely potential force, beyond Web development standards. In turn, more government regulation also suggests increasing litigation between vendors and consumers; the ongoing litigation between the U.S. Department of Justice and Microsoft exemplifies this potential. The growing popularity of open-source products, such as Linux, and the ongoing debate about software and intellectual property in the information economy also suggests regulation and litigation will be more of an issue in the software market in the near future.

Although the result is that the software product market is changing ISD, the existence of such a market does not mean the disappearance of custom ISD work. Many organizations may find no commercial products meet their needs. That is, while the market perspective applies to many forms of ISD, I want to highlight the market's emergence and resulting changes to ISD, not argue for the market's ubiquity.

A market-oriented perspective on ISD suggests consumer organizations will reconfigure their internal IT units. For example, software vendor issues, including how to influence functional requirements and align strategic plans, become important aspects of consumer organizations' technical leadership. Moreover, broader organizational aspects of software use, including organizational change management and the implications of that change, will even more important than in the past. For example, the long lag time between a user's request for a new system and the system's ultimate implementation—a hallmark of the build-oriented IS departments prior to the late 1980s—has now reversed. A market-based perspective on ISD changes the way software is interpreted and systems are developed. Software vendors today can produce new releases (often annually) faster than consumers can absorb them. C

References

1. Brooks, F. *The Mythical Man-Month.* Addison-Wesley, Reading, MA, 1974.
2. Carmel, E. American hegemony in packaged software trade and the "culture of software." *Info. Soc. 13,* 1 (Jan. 1997), 125–142.
3. Carmel, E. and Becker, S. A process model for packaged software development. *IEEE Transact. Engin. Mgmnt. 41,* 5 (1995), 50–56.
4. Carmel, E. and Sawyer, S. Packaged software development teams: What makes them different? *Infor. Tech. People 11,* 1 (Jan. 1998), 7–19.
5. Cusumano, M. and Selby, R. *Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People.* Free Press/Simon & Schuster, New York, 1995.
6. Cusumano, M. and Smith, S. Beyond the waterfall: Software development at Microsoft. In *Competing in the Age of Digital Convergence*, D. Yoffie, Ed. Harvard Business School Press, Boston, 1997, 371–411.
7. Keil, M. and Carmel, E. Customer-developer links in software development. *Commun. ACM 38,* 5 (May 1995), 33–44.
8. MacInnes, I. *Compatibility Standards, Market Power, and Antitrust Policy: Proprietary Interfaces and Their Alternatives.* Unpublished Ph.D. dissertation, University of Southern California, Los Angeles, 1998.
9. Royce, W. Managing the development of large software systems. In *Proceedings of IEEE WESCON*, 1970, 1–9.
10. Weinberg, G. *The Psychology of Computer Programming.* Van-Nostrand Rheinhold, New York, 1971.
11. White, H. Where do markets come from? *Amer. J. Soc. 87,* 3 (Sept. 1981), 517–547.
12. Zachary, G. *Showstopper: The Breakneck Race to Create Windows-NT and the Next Generation at Microsoft.* The Free Press, New York, 1994.

Steve Sawyer (sawyer@ist.psu.edu) is an associate professor in the School of Information Sciences and Technology at the Pennsylvania State University, State College, PA.