SUMMER 2001 VOL.42 NO.4

MITSloan
Management Review

Satish Nambisan

Why Service Businesses Are Not Product Businesses

REPRINT NUMBER 4247

Why Service Businesses Are Not Product Businesses

Although synergies exist between product and service sectors in most technology industries, extending beyond your dominant position requires understanding how the sectors differ.

Satish Nambisan

What manager in a product-oriented business has not thought about growing revenues by adding related services? What company in a service business has not weighed the advisability of offering products? Expansion often seems to make sense, but most companies creating a new offering limit themselves to approaches that are familiar and have worked before. Unfortunately, all too often those approaches are not suited to the new sector. The software industry provides examples that could help managers in other industries — particularly other high-technology industries — avoid getting injured in the chasm between service businesses and product businesses.

With the explosive growth of the global software industry, which witnessed annual revenues topping $500 billion in 1998, a striking phenomenon has emerged: software companies attempting to straddle the two industry sectors, products and services.[1] Traditionally, individual software companies catered to either the service market or the product market, but competition and the need to maintain a high growth rate induced many service companies to venture into the product arena.[2] To a lesser extent, product companies have started widening their offerings to include services.

It is too early to judge how successful product companies are in offering services, but the lackluster results in the opposite direction are evident.[3] A recent survey shows that from 1995 through 1998, approximately 87% of the product ventures initiated by software-service companies were unsuccessful.[4] Traditional explanations for their failures (such as lack of venture capital and marketing skills) are hard to accept, given the availability of capital and talent and the fact that Internet technologies have removed several location-based disadvantages in software development.[5] Instead, the failures point to more-fundamental issues.

It appears that many software-service companies that attempt to move into the product sector fail to recognize the differences between the two sectors' underlying business models. The implications for design and development practices are significant, and inappropriate transfer of organizational practices and development culture from the service sector to the product sector is a recipe for failure. A 1999 study of 21 software-service companies addressed the following questions: What current practices and strategies of a service company could inhibit its success in the product sector? What accommodations should be made to the software-development culture to support both product and service businesses? What organizational mechanisms should be deployed to capture potential synergies between the two businesses?[6] Although the study focused on the move from the service sector to the product sector, it also provided food for thought for companies making the move in the opposite direction.

The issues are not limited to the software industry. An increasing number of other high-technology industries (including computers, electronics and semiconductors)

Satish Nambisan *is an assistant professor of management at Rensselaer Polytechnic Institute's Lally School of Management and Technology. Contact him at nambis@rpi.edu.*

are witnessing a gradual blurring of the boundaries between their product and service sectors. Dell, Sony and others have embarked on ambitious projects to incorporate service offerings into their portfolios.[7] As the software-industry examples reveal, straddling the product and service sectors successfully requires the ability to understand the differences in the underlying economic and market forces affecting the two sectors — and the adoption of appropriate organizational practices.

Five Key Issues Differ for Products and Services

Software-product businesses and service businesses both involve software development and are global, knowledge-intensive and highly dynamic. However, they differ in several respects, including the economics of each business and how the businesses create value.

Intellectual Property Rights To the software-product vendor deciding the technology strategy and focus, legal protection of the product's special features is critical.[8] If strong protection is difficult to obtain, the only way to maintain competitiveness is to pursue rapid innovation and to bring out a continuous stream of upgrades that maintain the product's uniqueness (as Netscape did in its early years). For software-service companies, intellectual property rights are less important. Companies rarely own the rights to the services they develop or to the knowledge generated during development. Thus they lack the experience and processes to handle intellectual property rights appropriately for a product-development plan.

Product Complementarity Software-product companies expend a lot of effort developing products that complement, support or enhance the functionality of existing and established products. By leveraging the potential for products to complement one another, vendors can gain larger market size and make it more costly for customers to switch. For example, the initial success of i2 Technologies (a supply-chain solution provider) was due not only to its product's unique functionality, but also to how well the product complemented the leading enterprise resource planning (ERP) products. (ERP products provide companies with one comprehensive computer system that will integrate all the functions or tasks of the computer systems used by different departments or business units.) Software-service companies, on

the other hand, rarely concern themselves with the long-term competitive implications of which application area or software platform they choose to specialize in, because of the ad-hoc nature of the projects they undertake. More often than not, their focus is defined only by the range of software projects that they have managed to win over the years.

Returns From Scale Because fixed costs predominate in the development of products, companies can expect higher marginal returns from increases in market share (that is, from scale). The services sector, however, is dominated by variable costs, and therefore increasing returns from scale are rarely possible.[9] Achieving returns from scale would require extensive reuse of client specifications in other contexts — inherently difficult and often not contractually permissible. Such disparities in the two sectors lead to different behaviors: The promise of scale benefits may encourage product companies to make higher initial investments in the design of the product architecture. Service companies, for their part, handle variable costs by emphasizing process rigor and efficiency.

Abstracting Knowledge and Integrating Technology A product company's success lies in its ability to abstract the knowledge obtained while developing a product. Context-specific elements must be removed, so that the final product can be deployed in varied situations.[10] Service companies are in a different situation. Because their primary objective is to develop a system that is uniquely suited to the idiosyncrasies of the client's business, abstracting knowledge is not emphasized. Further, in the product sector, the varied technologies used in development must be integrated in the product architecture to enable seamless operation of the end product. Thus when integrating into its ERP product a technology developed by another company, SAP embeds it deeply in the architecture. Service firms, on the other hand, emphasize the ability to compartmentalize external technologies so that integration can be effected without reducing the efficiency of the overall development process. A Singapore-based service company integrated an external data-analysis tool into a claims-administration system that it was developing for an insurance company merely by creating a data bridge between the core system and the tool — without considering architectural compatibility. The approach may not result in optimal performance, but it definitely reduces development cost and time.

Comparing Product and Service Companies on Five Key Issues

Key Issue	Software-Product Companies	Software-Service Companies
Intellectual property rights	Very important	Less important
Product complementarity	Very important	Less important
Returns from scale	A fixed-cost structure allows for higher marginal returns from scale.	A variable-cost structure makes increased returns from scale rare.
Abstracting knowledge and integrating technology	The company must be able to capture generic product knowledge so that the product can be used in a variety of contexts. Architecture-level technology integration is important for the smooth running of the end product.	Knowing clients' idiosyncrasies is more important than knowledge abstraction. Companies rely upon data-interface-based technology integration; the primary emphasis is on development efficiency.
Connections with users	Companies have long-term relationships; typically, the users are technologically sophisticated.	Companies have project-driven relationships; typically, the users are technologically unsophisticated.

Connections With Users The knowledge derived from users shapes software development in both the product and service sectors. However, because the users themselves and the nature of their contributions differ, their relationships with companies differ — as do the ways companies manage those relationships. Users are a critical source of innovation for product vendors, who commit considerable resources to long-term relationships.[11] In the service sector, relationships with users are more project-driven and dominated by short-term goals. Further, because service companies often are involved in the development of mundane business applications, such as accounting systems and inventory control, their interactions are generally with unsophisticated users.

So although product and service companies may both be involved in software development, organizational practices differ. (See "Comparing Product and Service Companies on Five Key Issues.") Companies that have operated in either sector for long tend to emphasize the practices and culture they know best. When they cross the product/service divide and start operating in both sectors, the desire to find synergies often results in inappropriate transfer of development practices and methods — and leads to failures. For example, when a service company transfers its focus on process efficiency and rigor to products, those products may end up with limited design flexibility and create nightmares for long-term product support. The follow-

ing cases demonstrate how five software-service companies addressed the managerial challenges of moving into products.

Failure and Success in Moving from Services to Products

The following short case studies come from the author's 1999 study of 21 software-service companies. The companies, located in India, Singapore and the United States, pursued product initiatives to varied extents. The first three met with failure; the last two were successful.

Case One: When a Service Strength Becomes a Product Weakness Several years back, an India-based software-service company with expertise in developing accounting and financial systems for manufacturers decided to develop a generic accounting package targeted primarily at small and midsize manufacturers. Although the product achieved some initial success, by the end of the second year it had not captured sufficient market to justify further investments, and the company was forced to scale back. What went wrong?

First, the company based the product largely on the solutions it had created for its major clients over the years. Leveraging its internal expertise provided time and cost advantages, but the company failed to incorporate market elements to enhance those advantages. Its generic accounting product had too few unique features to differentiate it in the market. The company was counting on its domain expertise and its reputation in the service business to create sufficient market advantage and underestimated the significance of other elements that would have given the product a wider customer base. For example, it failed to integrate the product with leading products in complementary application areas (such as inventory control or taxation). Similarly, it failed to offer the product on multiple software platforms. The product manager admitted that the company's service solutions had been for clients who used the Unix platform, and that its expertise ended up limiting its product: "We prided ourselves so much on our [Unix-based] development expertise that we focused solely on that [platform] for product development." The lesson? Too much focus on leveraging its existing internal strengths made the company neglect other critical market elements that would have forced it to redefine the product's focus or even to sacrifice advantages such as Unix expertise in order to gain a more defensible market position.

The second factor in the product's failure also relates to development practices imported from the service business. Over the years, the company had put a premium on process rigor and efficiency. The downside was process inflexibility. On three separate occasions during the development of the first version of the product, project managers rejected demands for incorporating changes in the product specs, citing the need to remain efficient. As one later noted, "We didn't realize … until very late in the game that the one quality that was most desirable in our service [business] … had become to some extent a handicap in [developing] products." Given the dynamic nature of most product markets and the uncertainty regarding rapidly emerging competitive products, it is critical for product vendors to adopt a flexible and evolutionary product-definition process.[12] Thus, when service companies venture into the product sector, they must reexamine the trade-off between process flexibility and process rigor, and make the necessary changes.

Case Two: When Ease of Development Hampers Design Flexibility In the mid-1990s, a midsize software company based in Singapore, experienced in developing custom solutions for companies in the financial sector, decided to build a product to support the treasury- and securities-related operations of financial institutions. The new product found some early success. Unfortunately, at the end of the first year, the financial industry went through radical change, partly as a result of new government regulations, and the company had to revise its product to meet the requirements. In bringing out the new version, the company ran into technical difficulties, and the ensuing delays drove several potential customers away and hampered the product's growth. By the end of the second year, having revised only one part of the product, the company realized that the initial promise would never be realized.

The company could not respond rapidly to the new business requirements because, like many software-service businesses, it

Service companies are in a different situation. Because their primary objective is to develop a system that is uniquely suited to the idiosyncrasies of the client's business, abstracting knowledge is not emphasized.

An important challenge for a service company entering the product sector is to shift from a transaction orientation to a development orientation.

had sacrificed design flexibility to ease of development. Product architectures must be able to leverage market forces, even if that makes development more difficult. Modular product architectures enhance long-term product flexibility and maintainability, and cross-platform product design ensures broader market appeal. However, both modularity and cross-platform compatibility require more-complex designs and higher levels of knowledge abstraction, which may reduce short-term development efficiency. In the service sector, ease of development dictates much of the design strategy.[13] Given that profits are directly related to programmer productivity, simple system architectures that enable efficient development and coding are preferred to technically elegant and flexible architectures that require more development time.

The Singapore company's product architecture was difficult to modify when new requirements arose. The CEO later reflected that it would have been wise to specify criteria that could have alerted managers to the inappropriateness of some of the trade-offs the developers were making. Instead, the yardsticks that senior managers used to control the project (development time and cost, for instance) drove the developers to adopt practices and policies based on their original service orientation. In the end, the company faced a significant redesign and a rewrite of the code, a task that eroded all its initial advantages.

Case Three: The Perils of Overgeneralizing Some years back, a U.S.-based software company that specializes in custom-building decision-support systems (DSS) for companies in the financial industry decided to develop a generic DSS tool for financial-risk evaluation. It created a product targeted at managers in different industries who evaluate and control financial risk. Unfortunately, the product was a major market failure.

The company made a critical mistake while defining its product focus. In its service business, it had created unique, innovative features that were invaluable for its financial-industry clients. However, in choosing to target a wider market, the developers were forced to create a more generic product, forgoing innovative features that made little sense outside the financial industry. In failing to leverage its unique knowledge assets fully, the company created a product that was too generic to have any marketplace advantage. As one of the company's senior managers noted, "We had some unique things to offer and a good reputation with our financial clients … in hindsight, we should have developed only for that [niche market]." The niche strategy might have limited the product's appeal to a smaller customer base, but it would have allowed the product to retain its uniqueness.

Case Four: Successful Alliances With Other Product Vendors and Users A few years ago, a U.S.-based software company that develops custom logistics solutions for the chemical and pharmaceutical industries decided to create a product that would leverage its knowledge assets, including the powerful software-based analytical tools it had created to solve complex routing and scheduling problems for clients. However, an external market evaluation showed that a generic product might have difficulty finding a market as a stand-alone solution. So the company decided to piggyback on the logistics module of an established enterprise-resource-planning vendor serving the same industries and clients. As the company's CEO noted, "The crucial element of our success has been the selection of a complementary product that would add value to our end product and at the same time enable us to package and serve the knowledge built over years to an established customer base." Such an approach compensates for the company's use of relatively particularized knowledge (peculiar to the chemical and pharmaceutical industries) by aligning the product with an established product that would provide ready-made market-protection mechanisms in the form of an established customer base, brand equity and a stable technological platform.

The product's success was also partly due to the measures the company took to develop strong links with users. As the chief technology officer noted, "Our user network, which we have painstakingly built up over the last two years or so, is our lifeblood … it has enabled us to understand the real value of the knowledge our service business brought in and prevented us from making a number of wrong design choices." That emphasis on the user relationship is unusual for a service organization. Most service companies see their relationships with users as transaction-oriented (based on specific projects); in a transaction-oriented relationship, the established relational mechanisms

facilitate only limited information sharing. The product sector, however, calls for development-oriented user relationships that can support both knowledge sharing and knowledge generation. Thus, an important challenge for a service company entering the product sector is to shift from a transaction orientation to a development orientation. The shift is difficult because it relates to the organizational culture; it may need to be addressed at the level of individual developers, who will have to start viewing users not as one-time customers but as long-term partners.[14]

The company in question took specific measures early on, creating new roles (such as relationship manager) to cultivate selected service clients who could contribute to product development. The company also gradually expanded its user network by including its ERP partner's lead customers. To keep the users interested in participating in product development, the company offered incentives, including special implementation and integration services for the new product. The company strongly believes that the relationships were invaluable in the initial years of its product initiative and determined its eventual success.

Case Five: Successfully Using a Long-Term Design Perspective In the late 1980s, one of India's large software-service companies developed a branch-automation solution for a large Indian bank. Over the years, it added modules and also developed and implemented similar custom solutions for other clients in banking. Then, in the 1990s, the company developed a generic banking product based on its custom solutions. The product found success in both the Asian and European markets, thanks largely to practices the company adopted in product design and knowledge management.

The company realized early on that the key challenge would be to design a product capable of evolving. Creating a malleable architecture from custom-built software requires understand-

Managerial Challenges and Lessons Learned

1. Competitive Strategy

MANAGERIAL CHALLENGE: To define the focus of the service and the product businesses such that the knowledge assets acquired in each can be cross-leveraged.

1. Acknowledge the different ways in which the five key factors (intellectual property rights, product complementarity, scale benefits, abstracting knowledge and integrating technology, and connections with users) affect the two businesses.
2. Identify core domain themes or competencies that bridge the two businesses.
3. For service companies entering the product sector, consider the following strategies:
   a) Focus on a niche market and keep costs low while minimizing knowledge abstraction. (See cases three and four.)
   b) Complement and link with an established product while minimizing a focus on abstracting knowledge. (See case four.)
   c) Package generic domain knowledge if it possesses unique and defensible knowledge assets. (See cases one and five.)

2. Knowledge Sourcing

MANAGERIAL CHALLENGE: To manage external and internal links so as to support the underlying knowledge demands of both product and service businesses.

1. Acknowledge and address the differences in the ways that external and internal connections are maintained in the two businesses.
2. Establish shared knowledge-management systems based on specific themes or competencies that cross the product/service divide.
3. Deploy organizational mechanisms to manage external and internal connections:
   a) structural mechanisms (for example, relationship managers, as in case four)
   b) activity-oriented mechanisms (for example, cross-business process audits, as in case five)

3. Development Strategy

MANAGERIAL CHALLENGE: To balance design and development trade-offs in the manner most suited to the particular business (product or service).

1. Understand how the process of product design and development differs depending on the business (product or service) and recognize the implications of the key trade-offs:
   a) design flexibility vs. ease of development (See case two.)
   b) process flexibility vs. process rigor and efficiency (See cases one and two.)
2. Acknowledge that trade-off decisions are often a matter of deeply ingrained habit; counter by adopting appropriate design and process metrics.
3. Adopt a shared-process framework spanning the two businesses and use it to facilitate process learning as well as to guide companywide resource allocation.

Product companies are still behind on the learning curve when it comes to process efficiency and discipline. As one software manager noted, "It is very easy to bleed money on a custom [software-development] project."

ing which elements will be constant and which may change — and then designing the product around the major stable elements. To determine the extent of potential variability, the company classified each software component as vertical (engineered for a specific business domain) or horizontal (broader applicability), logical (providing business functionality) or physical (a basic technological component), and mandatory (core functionality) or optional (customers' choice).[15] By analyzing those three attributes, developers could assess the expected variability in each component and arrive at an architecture that would allow them to leverage the development work that had already been done in the service business while providing the flexibility to revise the product with ease. The design focus enabled the company to create gradually a product family from the initial custom solution, a feat that can provide considerable market benefits if done well.

The company's management of the connections among its employees was another important factor. Although product vendors frequently rotate their developers across different product groups to ensure knowledge sharing and generation of new ideas, service companies rarely have much need for information sharing across project groups. Many service companies intentionally keep their developers dedicated to specific application groups to gain economies of learning. As a result, service companies may lack mechanisms for establishing and promoting the internal relationships that facilitate knowledge sharing across project or product groups. This company, however, took several initiatives to establish effective mechanisms for speedy and efficient consolidation of expertise as well as for knowledge sharing across its product and service businesses. For example, it instituted regular project reviews and audits to identify experiential learning and created an organizationwide knowledge repository to store such knowledge assets. It also created a knowledge directory (a people/knowledge map) that pointed to expertise within the organization, using a multilevel taxonomy of topics that crossed the product-service divide. The company complemented those measures with development methodologies that called for more-intense interactions across groups, an approach that forced individual developers to establish stronger internal ties. Together, the measures enabled the company to bring about a gradual change in the nature and purpose of its internal links, and to leverage on an ongoing basis the development work carried out in its product and service businesses in related areas.

The cases illustrate the issues that can contribute to the success or failure of a service company's venture into the product arena. The issues can be classified into three interrelated categories: competitive strategy, knowledge sourcing and development strategy, but the key to success in all cases is to question ingrained development practices. Too often, companies not only fail to recognize that point, they mistakenly attribute their lack of success to external factors and end up making ever increasing resource commitments without effectively addressing the more fundamental problem of their development culture's biases and practices. (See "Managerial Challenges and Lessons Learned.")

Service Initiatives of Product Vendors

Software-product companies suffer from a similar blindness regarding biases in development practice and corporate culture when they attempt to incorporate software services into their portfolio. Product vendors have traditionally adopted an arms-length approach to services, typically offering services in partnership with established consulting companies. However, more recently, several product businesses, including SAP, Intuit, Oracle and i2 Technologies, have started establishing internal units exclusively to offer software services such as product customization and application hosting. For product enterprises, three issues are especially significant.

First there is the trade-off between process efficiency and flexibility. As they venture into the service sector, most product companies need to adopt process frameworks that impart more rigor into their development process, given the service sector's greater emphasis on process efficiency and productivity. The fact that product companies have lagged behind service companies in adopting process-improvement models such as the capability-maturity model (CMM) indicates that product companies are still behind on the learning curve when it comes to process efficiency and discipline. As one software manager noted, "It is very easy to bleed money on a custom [software-development] pro-

ject. … If you are not careful, you could easily end up paying for the inefficiencies of your client." Process models enable companies to identify and isolate such sources of inefficiency.

Second, although offering application-hosting services enables product companies to introduce new technologies and solutions to their clients rapidly, it also requires reexamining most practices with regard to design flexibility and product integration.[16] Further, such hosting services make clients more dependent on software companies, which may necessitate reevaluation of how the customer relationship is managed.

Third, product companies must institute whatever mechanisms are necessary to take full advantage of the potential synergies between the two sectors. For example, companies could use their product-customization and applications-hosting services (regardless of whether such services are offered directly or through an external partner) as an avenue to cultivate and evaluate their products' new functionalities and features. To do so, the product company would have to consider software service as an integral part of its product-development process rather than as just an extension of its customer support, as many product vendors currently view it. The new mind-set has important implications for the type of resources allocated to the service business as well as the practices and methods promoted in that business.

Implications for Other Industries

The issues and the managerial challenges discussed here are likely to be relevant to companies in other high-technology industries that have both a product face and a service face.[17] Examples include the semiconductor industry, in which service companies that design and produce customized specialty chips for companies are now branching out into the design and production of general-purpose chips, and the communications industry, in which service providers who traditionally have focused on designing and implementing customized networking products have started offering packaged networking solutions that cater to the general needs of a particular industry (for example, banking).[18]

As high-technology companies attempt to straddle both the product and the service sectors, some failures are inevitable — not from lack of expertise or lack of capital, but because of organizational practices and assumptions that are transferred from the dominant business. There are synergies between the product and service sectors in most technology industries, but the differences in the nature of their value creation and their delivery call for different types of design and development trade-offs and decision frameworks. Although the specific issues are likely to vary based on the industry context, the broad lessons learned in the software industry could become invaluable in avoiding potential disasters in other high-technology industries. The words of one interviewee, a CEO, go to the heart of the matter: "In our company there has been a constant tension between finding synergy between our product and service businesses and … recognizing the fundamental differences in the market conditions that drive the two businesses. … A significant part of our success in the last couple of years can be attributed to how we have managed the delicate balance between these two forces. … It has not been an easy task, and indeed, at many times, it has forced us to relearn and redefine our practices and policies."

ADDITIONAL RESOURCES

In addition to the references below, readers may be interested in a 1992 theoretical article written for the journal Economics of Innovation and New Technology by S. Torrisi and F. Malerba ("Internal Capabilities and External Networks in Innovative Activities"). David Mowery edited a 1996 Oxford University Press book that provides a good discussion of the global software industry ("The International Computer Software Industry").

R. Schware's discussion of the software-management issues faced by emerging countries is available in World Development 20, "Software Industry Entry Strategies for Developing Countries: A 'Walking on Two Legs' Proposition."

The reader also may find related information at several Web sites: the Software Industry Center at Carnegie Mellon University (www.heinz.cmu.edu/swic), the National Association of Software and Service Companies in India (www.nasscom.org), Electronic Business (www.eb-mag.com) and the France-based Organization for Economic Cooperation and Development Web site (www.oecd.org).

REFERENCES

1. "Software products" refers to packaged solutions that meet generic computing requirements and includes enterprise solutions (e.g., accounting systems or inventory-control systems), software-development tools, operating systems and utilities, and personal-computing tools. "Software services" refers to software-development and operations services provided to clients on a project basis and includes custom-software-development services and systems-implementation and systems-integration services.

2. R.B. Heeks, "India's Software Industry" (New Delhi: Sage Publications, 1996); S. Torrisi, "Industrial Organization and Innovation: An International Study of the Software Industry" (London: Edward Elgar, 1998); A. Lateef, "Linking Up With the Global Economy: A Case Study of the Software Industry," International Labor Organization report no. 96 (1997); D.J. Hoch, C.R. Roeding, G. Purkert and S.K. Lindner, "Secrets of Software Success" (Boston: Harvard Business School Press, 1999); "Irish Software Industry Survey" (Dublin: National Software Directorate, 1998); and NASSCOM-McKinsey report on the Indian software industry (December 1999), http://www.nasscom.org.

3. H.V. Sukhathankar, "Time To Ponder: The Indian Software Industry," Dataquest India, July 30, 1997, 21-24; and A. Arora, V.S. Arunachalam, J. Asundi and R. Fernandes, "The Indian Software Services Industry," working paper 99/19, Carnegie Mellon University, Heinz School of

Public Policy and Management, Pittsburgh, Pennsylvania, December 1999; and K. Ryan, "Irish Software Industry — Lessons Learned" (proceedings of the PICMET Conference, Portland, Oregon, July 1997).

4. The author's 1999 questionnaire-based survey involved 134 software service companies (73 in India, 42 in the United States and 19 in Singapore) that had initiated a product agenda in the prior three years. The survey data showed that 52% dropped further investment in the product after the first year.

5. For a discussion of the changing market forces in the global software industry, see C. Anderson, "The Software Industry: The Birth of a New Species," The Economist, May 25, 1996, 35-53; see also "Forecasting a Robust Future: An Economic Study of the U.S. Software Industry" (New York: Business Software Alliance, June 1999); and R.B. Heeks, "Software Strategies in Developing Countries," Communications of the ACM 42 (June 1999): 15-20.

6. The interview-based field study involved large and small software-service companies in India (10), the United States (7) and Singapore (4). All were pursuing product agendas to varied extents. Interviews were conducted with senior and midlevel managers. A total of 49 unstructured interviews were conducted over a six-month period; each interview lasted about an hour and covered such issues as interviewees' business strategies in the two software sectors and the changes in their organizational structure, development practices and processes, and knowledge-management practices. White papers and other published documents supplemented the interview data. The 10 Indian companies were, on average, 8 years old, had average annual revenues of $21 million (7% from products alone) and 139 employees; the seven U.S. companies had an average age of 11 years, average annual revenues of $32 million (11% from products alone) and 148 employees; the four Singapore companies had an average age of 6 years, average revenues of $3 million (10% from products alone) and 53 employees.

7. In February 2000, Dell Computer initiated Dell E-Works, which offers enterprise computer systems and storage products in the form of services to clients. Similarly, Sony's new business unit, Broadband Services Co., has a mandate to exploit all Sony's hardware, systems solutions and other technology assets to create new service businesses.

8. Economists term the conditions affecting the probability that an innovator can reap the benefits of its own innovation the *appropriability regime*. Torrisi, "Industrial Organization," 121, discusses how it relates to the global software industry.

9. Heeks, "India's Software Industry," 1-45.

10. Hoch, "Secrets of Software Success," 38-47.

11. Indeed, the decline of several large European software-product vendors in the 1980s has been attributed to their lack of exposure to sophisticated users, and hence, their inability to build up capabilities to compete in the global software market.

12. M. Ianisiti and A. MacCormack, "Developing Products on Internet Time," Harvard Business Review 75 (September-October 1997): 108-117; S. Bhattacharya, V. Krishnan and V. Mahajan, "Managing New Product Definition in Highly Dynamic Environments," Management Science 44 (November 1998): 50-64; and M. Cusumano and D.B. Yoffie, "Competing on Internet Time" (New York: Free Press, 1998).

13. Torrisi, "Industrial Organization," 91-128.

14. L.L. Constantine, "Constantine on Peopleware" (New York: Prentice Hall, 1995); and S.E. Donaldson and S.G. Siegel, "Cultivating Successful Software Development: A Practitioner's View" (New York: Prentice Hall, 1997).

15. For a discussion on using those attributes to assess potential variability in a product domain, see R. Ramaswamy and U. Nerukar, "Creating Malleable Architectures for Application Software Product Families" (proceedings of the Workshop on Object Technology for Product-Line Architectures, European Conference on Object Oriented Programming, Lisbon, Portugal, June 1999); also see J. Meekel, T.B. Horton and C. Mellon, "Architecting for Domain Variability" in "Proceedings of the Second International Workshop on Development and Evolution of Software Architectures for Product Families" (Berlin: Springer Verlag, 1998).

16. Although a few software-product companies such as SAP, Intuit, PeopleSoft and Oracle have taken the initial steps to offer application-hosting services directly to their clients, most other product companies have a wait-and-see attitude, acknowledging the difficulty of transitioning from the product to the service sector ("Software Shakeout," Business Week, March 5, 2001, 72-80).

17. In "Where Value Lives in a Networked World," Harvard Business Review 79 (January-February 2001): 79-86, M. Sawhney and D. Parikh use the concept of the migration of network intelligence to show how the boundaries between the product and the service sectors in various industries are reshaping the competitive landscape.

18. D. Delano, ed., "Global Semiconductor Industry Report," Electronic Business (1998); and K. Reeder, "Radical Changes in the Communications Industry: A Report" (Paris: Organization for Economic Cooperation and Development, 1998).

Reprint 4247

MITSloan
Management Review

Reprints/Back Issues

Electronic copies of SMR articles can be purchased on our website:
www.mit-smr.com

To order bulk copies of SMR reprints, or to request a free copy of our reprint index, contact:

MIT Sloan
Management Review
Reprints
E60-100
77 Massachusetts Avenue
Cambridge MA 02139-4307
Telephone: 617-253-7170
Toll-free in US or
Canada: 877-727-7170
Fax: 617-258-9739
E-mail: smr@mit.edu

Copyright Permission