Developing Software-as-a-Service in the Rapidly Changing Environment

Masao KAKIHARA*

Abstract

This paper explores Software-as-a-Service (SaaS), an emerging model of the software business in the Age of the Internet, from a strategic management perspective. Software development is now faced with two dynamic innovation streams: technological innovation and market innovation. Harshly shook by rapid technological development and highly volatile market environment, today's software development is under the constant necessity for swift and reliable development practices and market launch in appropriate timing. Software development, especially on the SaaS model, has to be more and more strategic. The paper suggests a strategic management framework for SaaS development in rapidly changing environments. Through a brief case study of online community services in Japan, the paper also suggests that dual roles of beta versions, as a product and media, would play a critical role in making strategic decisions in SaaS development in rapidly changing environments.

Keywords:
Software Development, Strategic Management, Software-as-a-Service

I. Introduction

All aspects of our social lives are now greatly dependent upon various kinds of software. Ranging from common application software like spread-sheet to large information systems in banking networks, software has become an irreplaceably critical part of our social systems. It is obvious, however, that today's software is now faced with rapid social transformation. For software engineering, whose traditional research question has been how to technically build reliable software, there is a strong need to offer social and managerial accounts of software development practices. In short, software engineering must explain not only 'how efficient' but also 'how effective' the developing practices are in a given social and managerial context. Thus software engineering is no longer a 'closed' research field staying within a highly technical domain but must rather be 'open,' incorporating with broad research achievements in social sciences such as economics, management, sociology, and so on.

* Masao Kakihara is Assistant Professor, School of Business Administration, Kwansei Gakuin University. Email: kakihara@kwansei.ac.jp

This paper explores an emerging model of the software business; namely, Software-as-a-Service (SaaS). SaaS is a business model of delivering software functions and services to customers via the World Wide Web (WWW). In the SaaS model, customers can use the functions and services on demand and remotely through the Internet access. Nowadays, even package software is constantly updated and modified by program patches distributed from vendors via the Internet. This web-based software delivery model is now urging us to reconsider traditional software development frameworks and approaches. In today's turbulent business environment, software development is not an isolated practice confined only in technical fields; *any* software development needs to consider how to adapt to such environmental changes and to launch into an appropriate market in an appropriate timing. Furthermore, managers have to consider not just whether software projects are 'done properly' but also whether produced software makes expected results and returns. To put it simply, today's software development has to be more and more *strategic*. The more dynamic and turbulent environmental changes facing software development are, the more strategic the development practices have to be to cope with those changes efficiently and effectively. Such strategic management is particularly important for the software business on the SaaS model since the model drastically opens up software development practices toward dynamic interaction with such rapidly changing environments.

The paper is structured as follows. *Section 2* offers an outline of environmental changes with which software development is now faced in the Age of the Internet. *Section 3* discusses the concept of SaaS and its implications. *Section 4* briefly reviews strategic management approaches and *Section 5* particularly focuses on Eisenhardt's framework of 'Strategy as Simple Rules.' *Section 6* discusses a case study of online community services in Japan in relation to the SaaS model. Finally, *Section 7* summarizes the arguments and discusses limitations.

II. Software Development in the Age of the Internet

Given its ubiquity in our social lives, software is now functioning as an important actor in our society. In the dawn of the Computing Age, roughly from the 1940s to the 1950s, software was a problem domain inseparably woven into hardware issues. Through commercialization of computers in the 1960s, when epoch-making IBM's System/360 was launched, people had gradually realized the existence of software. Then, in 1975, Microsoft, the largest software company in the world, was founded by Bill Gates and Paul Allen. Even at that time, few could imagine that the software company would have succeeded tremendously in scale and social impact. In concert with Microsoft's gigantic success, we have seen during the last three decades the dramatic evolution of software as research and business fields.

During the last three decades, software has experienced a wide range of environmental changes, which can be summarized as follows:

1. *Rapid evolution of hardware*

Hardware systems have evolved rapidly: rapid increase of CPU performance, sharp decline of price of memory and storage devices such as HDD, diversification of input/output devices, etc. Moore's Law, the empirical observation that the complexity of an integrated circuit, with respect to minimum component cost, will double in about 18 months, seems still continue, and such rapid evolution of hardware inevitably affects the ways of developing and operating software.

2. *Widespread usage of software in society*

It is since the 1980s that computers have been used not only in quite limited situations such as research laboratories and governments but also in ordinary people's social lives. Personal computers have been diffused in households and people use a variety of software in their PC. Moreover, software systems embedded in electronic appliances such as mobile phones and TV displays are required to perform stably and correctly in diverse contexts, from seas to deserts. Such widespread usage of software inevitably demands unprecedented levels of quality and reliability of software.

3. *Diversification of stakeholders in software development*

The scale of software development projects has become larger than ever, and this resulted in rapid diversification of stakeholders in the projects. For example, whereas the number of lines of the source code of the *Windows 3.1* launched in 1992 is about 3 million, that of *Windows 2000* launched in 2000 has been increased into 35-60 million (METI, 2004). In order to cope with such a rapid increase in scale and complexity of software development, recent software engineering research has invented component-based development approaches facilitating more distributed development practices across organizational boundaries (Heineman and Councill, 2001). As a result, diverse stakeholders can participate in a particular software development project.

4. *Continuous revision of software functions*

Today's software can and has to evolve even after its market launch. Rapid and continuous changes of software usage as described above have brought forth an increasing demand for continuous revision of software systems. Diffusion of the Internet has further energized this trend, enabling post-launch distribution of program patches for update. Even package software is not be 'finished,' frequently revised by update patches via the Web. Such never-ending revision processes create new triggers for malfunctions and makes ROI evaluation of software development projects more and more difficult.

To summarize these rapid environmental changes, software development in the Age of the Internet must take account of two innovation streams: *technological innovation* and *market innovation* (Kakihara, 2005). Based on these two streams, three schematic models of software innovation can be identified (see Figure 1).

The first model of software innovation is one that actively takes full advantage of functional capability enabled by rapid technological innovation for new software development. Here it could be called the '*technology-push*' model. As discussed above, the pace of technological innovations is still quite rapid. Only within a year, many technological impossibilities turn to be possible thanks to new technological innovations. Technological innovations are not just limited to hardware innovations such as increase of CPU power and network speed but can be also software innovations such as new programming language and supporting programming techniques. Strongly 'pushed' by these innovations, functionalities and quality of software will be continuously innovated as well.

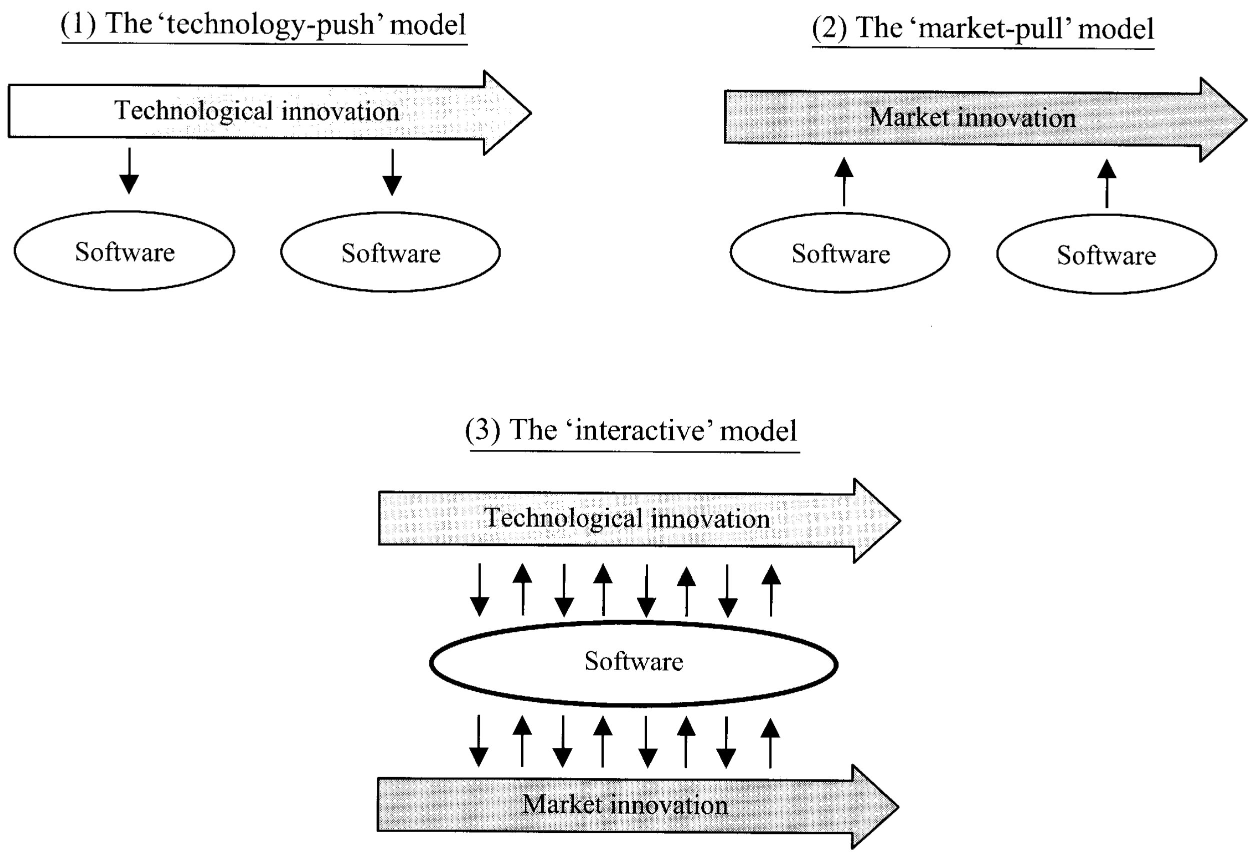

Figure 1: Three schematic models of software innovation
(Kakihara, 2005)

The second model is one that software innovation is 'pulled' by changing market environments such as demand levels and user preferences. This could be called the '*market-pull*' model. In general, any software is developed for fulfillment of certain market needs. For instance, spread-sheet applications are made to meet user's needs of efficiency and convenience in data handling in various business scenes. Likewise, embedded software installed in mobile phones enables much higher and more complex functionalities than a simple talking function. Strongly 'pulled' by such market demands, software development practices will also be innovated.

In reality these two schematic models of software innovation are not discrete or antithetical but rather interrelated and interactive. Software development in reality should be conducted in dynamic interaction with both technological and market innovations. This is the third model which can be called the '*interactive*' model. As the broad-band Internet access has been widely diffused, the interaction with technology and market has become much more dynamic and intense than ever. As a supply chain and communication media, the Internet can connect software vendors to a wide range of stakeholders, particularly other business partners and their customers. Unprecedented forms of collaboration that support software development practices have become possible. Today's software development can and should dynamically adapt its processes and frameworks to both innovation streams. Such a 'dialogue' with both environmental changes would be critical for today's software innovation.

For software development in the Age of the Internet, it is critically important to realize the fact that software development is no longer a craft isolated from the real-world and done by 'geeks' but rather has now become highly strategic conduct under the intense pressure of rapidly changing environments. Software development is now in the midst of dynamic environmental changes induced by rapid technological and market innovations.

III. Software-as-a-Service: An Emerging Model

This rise of the Internet in the software business is bringing forth a radically new trend: *service-orientation* of software.

The diffusion of the Internet is radically transforming the ways of distributing, selling, and maintaining software. In pre-Internet time, software, a set of digital codes, was dealt with as if it had been a physical product. In order to distribute it through the existing supply chains and exchange it in price-based market mechanisms, the digital codes had to be molded into some physical media such as floppy disks, CD, and DVD. Such institutional bottlenecks of the market systems forced software to be 'physical.' The diffusion of the broad-band Internet since the late 1990s transformed it dramatically. On the Web, the digital codes become freed. Since the broad-band Internet access is becoming ubiquitous in our social lives, software, in theory, can be distributed, sold, and maintained through the Web. Software no longer has to cling to the traditional market institutions built for physical products.

This clearly means that the software business is transforming itself from ‘product-based’ to ‘service-based.’ The vital point for today’s software users is not how to ‘own’ it but to ‘use’ it. In the ubiquitous information environments, digital goods become from possessed products to contract-based services like car rental. Looking at this service-orientation of the software business, Rust and Kannan (2003) argue:

> In changing the software product to a rentable software service, firms are forced to understand how the customer uses a piece of software. The design of the software becomes more customer-centric. By providing a software service in addition to selling it as a product, the firm learns more about the usage of its software and becomes more attuned to the needs of the customer, which contributes toward a competitive advantage. (p. 40)

This type of web-based software distribution is not new at all. The Application Service Provider (ASP) model has been used since 1990s particularly by the business service operators. However, the bandwidth of the Internet access at the time was quite limited for effective usage of the software service on the ASP model. Nowadays, the broad-band Internet access enabled by xDSL and optical fiber technologies have made the service-oriented software model truly ‘working’ in real business settings where a vast amount of data is constantly transacted.

This technological innovation especially in networking technologies transformed the somewhat outdated concept of ASP into a new concept with wider implications: *Software-as-a-Service* (SaaS). IDC (2005) defines the key characteristics of SaaS as follows:

- Network-based access to, and management of, commercially available (e.g., not custom) software
- Activities that are managed from central locations rather than at each customer’s site, enabling customers to access applications remotely via the Web
- Application delivery that typically is closer to a one-to-many model (single instance, multi-tenant architecture) than to a one-to-one model, including architecture, pricing, partnering, and management characteristics

SaaS is not just a technological option that enables a firm to deliver software to its customers; it is a broader concept of business model that interactively connects software venders to customers on the ubiquitous information environment. It frees the software business from traditional constraints rooted in the physical-based, possession-oriented market mechanism. IDC points out that there are two types of provider in the SaaS model: *hosted* *application management* and *software on demand.* The former is based on the ASP model, making software available to subscribed customers through the Web for normally monthly fees. The latter is a model that provides customized functions of software service with customers on a one-to-many basis. IDC also states that there are three underlying trends supporting the SaaS model: *from traditional license to subscription*; *from one-to-few to one-to-many*; and *from private infrastructure to public infrastructure.* These are all enabled by the diffusion of the Internet, a flexible supply chain and interactive communication media for virtually everyone (see Figure 2).

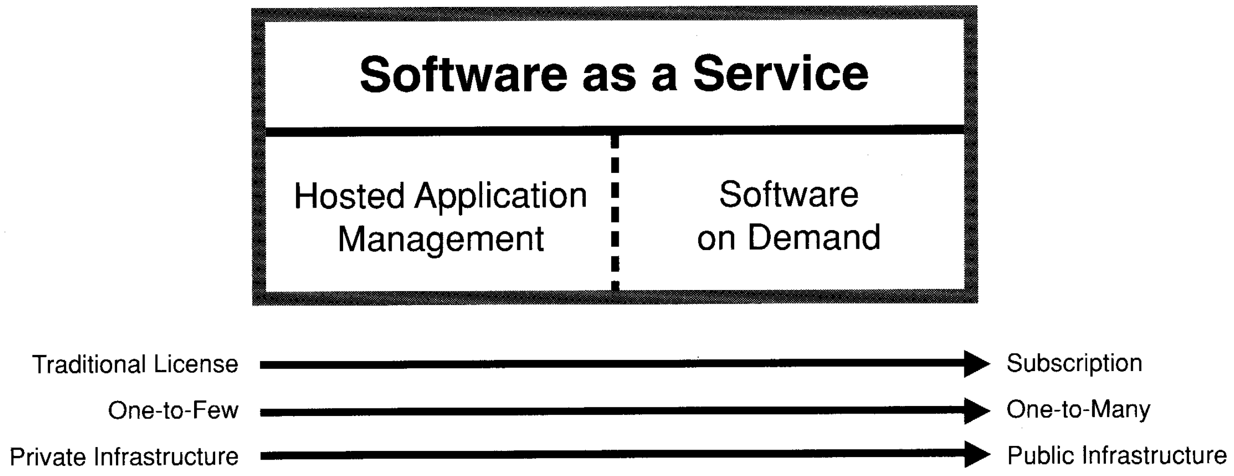

Figure 2: Software-as-a-Service coverage
(IDC, 2005)

IV. Strategic Management of Service-oriented Software Development

It is obvious that the emergence of SaaS as a new software business model inevitably changes the ways of developing software. In short, development of service-oriented software needs much more strategic approaches than ever, since it has to be conducted in dynamic interaction with technological and market innovations.

Traditionally, formal processes of software development are structured from 'requirements analysis' to 'architecture and design' to 'coding' to 'test' to 'deployment and maintenance.' This process model is generally labeled the 'waterfall' model. Based on this linear process model, many modified models have been devised, such as 'incremental', 'spiral', 'concurrent', and 'evolutional' models (Pressman, 2004). However, the waterfall model is still widely used as a reference model in real software development practices mainly because sequential ordering of the development phases and inhibition of backtrack are particularly convenient for project management (Tamai, 1993).

In software engineering research, it has been often argued that strategically important is 'up-front' phases in the development process, namely, 'requirements analysis, and 'architecture and design.' These 'up-front' phases are the 'human-intensive' processes in which managerial and business intent and technological possibilities are strategically negotiated to define core features of developed software (Ibid.). It is also often argued that the more dynamic and ambiguous the environment in which software is launched is, the more detailed and deliberate the 'up-front' phases of software development should be in order to decrease operational risks in the lower phases.

This 'Big Design Up Front' approach, however, is often criticized since the approach would be quite unrealistic particularly for software development in rapidly changing environments. Rapid environmental changes inevitably create unexpected changes in requirements for software in the course of development processes. Given the above discussion of software innovation models, software development processes are constantly faced with two rapid innovation streams: technological innovation and market innovation. New technological innovations can drastically change technological assumptions for software development. Likewise, new market innovations can greatly transform targets to be reached by developed software. These rapid and constant environmental changes make 'up-front' planning and design seriously difficult or even problematic. McConnell (2004) is one of the main proponents who criticize such a 'Big Design Up Front' approach. He stresses the importance of 'construction' of software, rather than planning or design. He argues that software construction, focusing on coding and debugging, is "the central activity of software development" (p. 7). Supported by this kind of argument, software engineering researchers have creates more flexible and adaptable development approaches and models such as 'agile' and 'adaptive' software development (Cockburn, 2001; Highsmith, 1999).

Yet it can be argued that these recent software development approaches and models put 'too much' emphasis upon technical skills of coding and debugging and largely ignore the strategic importance of planning and design. To be sure, the 'Big Design Up Front' approach is in fact problematic in rapidly changing environments and it is understandable that recent software engineering research is pursuing cultivation of feasible construction skills for adaptive development for efficient and effective software development. However, this over-reaction against 'Big Design Up Front' is also problematic since it tends to result in the praise of 'No Design Up Front.' Such an all-or-nothing discussion would be quite unproductive for fruitful development of software engineering research.

As discussed above, strategic management of software development is increasingly crucial for efficient and effective management of software development. The diffusion of the Internet has opened up ever-isolated software development practices into dynamic interaction with technological and market innovation streams. Furthermore, the Internet has also facilitated service-orientation of software, transforming the software business from product-based to service-based. In particular, software businesses on the SaaS model are driven in a highly customer-centric manner. In such a dynamic environment, a 'good' strategy for software development would be located in the realm between two extremes: 'excessive design' and 'no design.' What software development in rapidly changing environments needs is a strategic framework that link design, construction, and implementation together to cope with such dynamic environmental changes. And so far, software engineering research poorly refers to vast research achievements of business studies in general and the strategic management field in particular. Whereas some management scholars are studying software development practices from a strategic perspective (e.g. Cusumano, 2004; Cusumano and Selby, 1995; MacCormack, 2001), research endeavor of software engineering research looking at strategic management issues is scarce.

V. Strategy as Simple Rules for Software Development

This section offers a brief outline of recent achievements of strategic management research and tries to apply them to software development practices.

Strategic management as a research field is one of the youngest among business and management studies. There must be little doubt on the opinion that it is Porter's study on competitive strategy (Porter, 1980; Porter, 1985) that first systematized various strategic management issues for modern firms and started the contemporary strategic management research. Based on industrial organization economics, Porter argues that firms need to realize *industry structure* that determines the profitability for a firm in the industry and hence the firm's competitive strategy. Based on the analysis of industry structure, firms need, he argues, to determine their *unique strategic positioning* to gain competitive advantages.

There has been some criticism to such a position-based framework of strategic management. Porter's framework rests upon the S-C-P (Structure-Conduct-Performance) paradigm of industrial organization economics, which demonstrates that a firm's competitive advantage is determined *a priori* by 'structure,' an outside environment that surrounds the firm. However, some scholars critically argue that a firm's competitiveness can be also determined by unique resources held inside the firm. Thus firms' *resources*, such as unique technological advantages and tacit organizational capability and knowledge, are gradually focused as a source of sustainable competitive advantage. Barney (2002), one of the main proponents of this *Resource-Based View* (RBV) of the firm, proposes the VRIO framework, namely, *value*, *rarity*, *imitability*, and *organization*, to systematically analyze capability of the firm.

Although many other strategy frameworks and approaches have been proposed so far

(Mintzberg et al., 1998), these two frameworks, position-based and resource-based, are widely accepted as the mainstream of the contemporary strategic management research. However, both frameworks are apparently questionable in applicability to software development, particularly that for service-oriented software. That is to say, the presupposed time-scale in strategic decision making in both frameworks is extremely dull. Both frameworks assume a relatively static environment for strategy settings, being with slow technological and market innovations and stable industry structure. Today's business environments, however, are so dynamic and shaky that no one can foresee them even half a year ahead. Harshly shook by rapid technological development and highly volatile market environment, today's software development is under the constant necessity for swift and reliable development practices and market launch in appropriate timing. Given these dynamic settings, the time-scale that the position-based and the resource-based frameworks presuppose is too coarse to make swift and sound strategic decisions. In software development practices in rapidly changing environments, there is virtually no time margin to determine strategic positioning against potential and/or existing competitors or to build unique resources that can serve as effective barriers for potential new entrants.

Here, there is another approach worth giving careful consideration in this context: Eisenhardt and her colleagues' study on *competitive strategy in high-velocity markets* (Eisenhardt, 1989; Eisenhardt and Brown, 1998; Eisenhardt and Sull, 2001; Eisenhardt and Tabrizi, 1995). She has been focusing on business strategy and product development in rapidly changing environments, especially the computer industry and the internet business.

Her recent study (Eisenhardt and Sull, 2001) particularly looks at strategic management in the internet business with a case study of Yahoo!. Yahoo! is without doubt one of the most successful service-oriented internet companies since the late 1990s. Starting as a portal site, Yahoo! is continuously developing and launching many novel services such as Yahoo! Calendar, Yahoo! Messenger, My Yahoo!, Yahoo! Music, Yahoo! Shopping, and many more. Analyzing Yahoo!'s success from position-based and resource-based frameworks of strategic management, she argues that both cannot offer firm reasoning.

> Everyone recognizes the unprecedented success of Yahoo!, but it's not easily explained using traditional thinking about competitive strategy. Yahoo!'s rise can't be attributed to an attractive industry structure, for example. In fact, the Internet portal space is a strategist's worst nightmare: it's characterized by intense rivalries, instant imitators, and customers who refuse to pay a cent. Worse yet, there are few barriers to entry. Nor is it possible to attribute Yahoo!'s success to unique or valuable resources – its founders had little more than a computer and a great idea when they started the company. As for strategy, many analysts would say it's not clear that Yahoo! even has one. ..... If Yahoo!

has a strategy, it would be very hard to pin down using traditional, textbook notions. (p. 108)

Based on this analysis, she argues that Yahoo! has a distinct strategy for competition in high-velocity markets; namely, *Strategy as Simple Rules*. The essence of this strategy is in the sentence "when business becomes complicated, strategy should be simple" (p. 116). Whereas the strategic logic of the position-based approach is "establish position" and that of the resource-based approach is "leverage resources," the main implication of strategy as simple rules is the importance of capturing "unanticipated, fleeting opportunities" (see Table 1). Through dozens of case studies on companies in fast-moving markets, she and her colleagues identify five simple rules that guide core strategic processes:

- *How-to rules*: Spelling out key features of how a process is executed
- *Boundary rules*: Focusing managers on which opportunities can be pursued and which are outside the pale

Table 1: Three approaches to strategy
(Eisenhardt and Sull, 2001)

	Position	**Resources**	**Simple rules**
Strategic logic	Establish position	Leverage resources	Pursue opportunities
Strategic steps	Identify an attractive market; Locate a defensible position; Fortify and defend	Establish a vision; Build resources; Leverage across markets	Jump into the confusion; Keep moving; Seize opportunities; Finish strong
Strategic question	Where should we be?	What should we be?	How should we proceed?
Source of advantage	Unique, valuable position with tightly integrated activity system	Unique, valuable, inimitable resources	Key processes and unique simple rules
Works best in	Slowly changing, well-structured markets	Moderately changing, well-structured markets	Rapidly changing, ambiguous markets
Duration of advantage	Sustained	Sustained	Unpredictable
Risk	It will be too difficult to alter position as conditions change	Company will be too slow to build new resources as conditions change	Managers will be too tentative in executing on promising opportunities
Performance goal	Profitability	Long-term dominance	Growth

- *Priority rules*: helping managers rank the accepted opportunities
- *Timing rules*: synchronizing managers with the pace of emerging opportunities and other parts of the company
- *Exist rules*: helping managers decide when to pull out of yesterday's opportunities

Yahoo!, for example, clearly holds four simple rules in developing and executing their services: 1) know the priority rank of each product in development, 2) ensure that every engineer can work on every project, 3) maintain the Yahoo! look in the user interface, and 4) launch products quietly. Sticking to these simple rules, Yahoo! succeeded in growing and surviving in the highly turbulent internet business market. The case of Yahoo! tells us that in high-velocity markets, firms must quickly find out unanticipated, fleeting opportunities and shrewdly seize them to grow faster than competitors. And yet they should not act in disorder. They must focus their strategy upon several important simple rules that swiftly guide significant business processes.

Each of the three frameworks of strategic management discussed above, has its own strengths and weaknesses. It would be clear that for developing service-oriented software, which constantly interact with rapid flux of technological and market innovations, the framework of strategy as simple rule can work more effectively. Moreover, the framework can also provide us with guiding principles for the 'agile' and 'adaptive' software development models, which well balance development practices between 'excessive design' and 'no design.'

VI. A Case Study: Online Community Services in Japan

This section discusses a brief case study of three online community services in Japan. Through analysis of the services, a potential framework for SaaS development is addressed based on Eisenhardt's strategy as simple rules.

A variety of services on the SaaS model have been already launched and they are normally used on web browsers without installing particular software packages. On the B2C (business-to-consumer) side, portals, search engines, e-commerce platforms, auction services, banking agents, all are now converging to the realm of SaaS. On the B2B (business-to-business) side, the traditional ASP model is transforming itself into the SaaS model particularly in Supply Chain Management (SCM) and Customer Relationship Management (CRM) fields. Although the SaaS model has been normally discussed in B2B contexts so far, three cases of SaaS in B2C, or rather to say C2C (consumer-to-consumer) contexts are particularly taken here. The reason is that, as discussed above, the essence of SaaS is its customer-centricity. Development and operation of SaaS is driven by dynamic interaction with customers that bring the firm valuable feedback to revise and update the service. In this sense, B2C/C2C would be a more

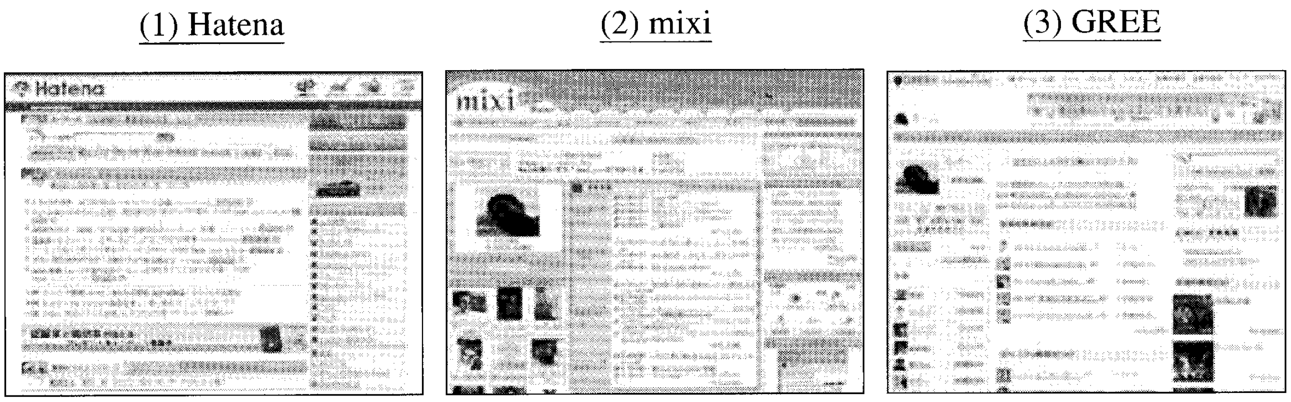

Figure 3: Online community services in Japan

appropriate context for the analysis of SaaS than B2B.

Three cases to be taken here are all online community services developed and operated in Japan: *Hatena*, *mixi*, and *GREE* (see Figure 3).

Hatena,[1] starting in 2001, is a set of online services that help users search, gather, and organize a wide range of information on the Web. It consists of Hatena Search, Hatena Diary, Hatena Antenna, Hatena RSS, Hatena Bookmarks, etc. The most fascinating feature of Hatena's service would be its development policy: *developing services together with users.* Hatena employs just about 15 members of staff including part-time. Mr. Junya Kondo, CEO of Hatena Inc., states that "[we] release our new service in a semi-finalized form and continuously revise and update them through a dialogue with users."[2] By taking full advantage of hundreds of thousands of users' knowledge, Hatena can develop and revise their services efficiently and effectively. Mr. Kondo also argues that "In Hatena's services, there is no distinction between prototypes and final products, and it can be seen both as ever-lasting prototypes and as final products from the first moment of market launch."[3] In fact, Hatena is perhaps the first provider of social bookmark service, Hatena Bookmark. This service has been frequently revised based on user feedback and, as a result, quickly become the *de facto* standard in this domain in Japan.

mixi[4] is the largest social networking service (SNS) in Japan. Just like other popular SNSs, mixi is a semi-closed SNS whereby only the existing users can invite new users. There are a vast number of user communities for specific topics. Since its service launch in February 2004, it has gathered over 2.6 million subscribed users just in two years. mixi's dramatic

1 http://www.hatena.ne.jp/
2 http://blog.japan.cnet.com/kondo/archives/002446.html
3 http://blog.japan.cnet.com/kondo/archives/002408.html
4 http://mixi.jp/

expansion of its user base resulted not only from its highly user-friendly interface but also the pursuit of a better communication that makes users stayed in the site for a longer time. Mr. Kenji Kasahara, CEO of mixi Inc., says that "There are three important points to plan our new service: better communication, usability in gathering interesting information, and expanding human networks."[5] All these points relate to facilitation of communication among users. In this sense, mixi's core policy of service development is user-communication-centricity.

GREE,[6] is another successful SNS in Japan. Mr. Yoshikazu Tanaka started his own SNS, named GREE, in February 2004 as a completely private project. Starting with just 4 users at its launch, it gathered over 10 thousands registered users only within a month. As of July 2005, the user base expanded into more than 200 thousands. Currently, GREE is operated by a company, GREE Inc., that Mr. Tanaka founded in December 2004. From a strategic point of view, the most distinctive characteristic of GREE's service development is that when they upgraded the alpha version of the service in October 2005, they defined their service as the *perpetual beta*. Mr. Tanaka said[7] that "We are actively taking advantage of various new technologies by continuously caring about user benefits and usability." He as the founder also stated that in order to seize unexpectedly rising opportunities in the field, "swift decision making moment by moment is crucial." He also said that he always paid great attention to "efficiency in decision making" and that "short time-span for launch of new services and/or functions" holds great impacts upon the whole operation. Moreover, in order to operate the service so swiftly, he insisted that "adaptive organizational structure against sudden environmental changes" is of paramount importance.

The brief case study of the three online community services tells us that all of them have been developed and operated in quite simple rules: launch new services swiftly and listen users' voices. This is exactly the strategy as simple rules. In rapidly changing environments, 'big design' or 'big plan' does not work well. Rather, the desired strategy of service development and operation would be a simple and adaptive strategy that quickly tailors the services to user requirements.

In such a simple and adaptive strategy, there is a unique feature that is strikingly different from traditional package software model; namely, *the dual roles of beta versions*. The original purpose of releasing beta versions is to evaluate and finalize the software's usability and functions and to make final bug-fixing smooth through feedback from beta testers. For SaaS, which will never be finalized, a beta version is not only an actual *product* delivered to users but also *media* through which the next beta version is devised with dynamic negotiation with

[5] http://japan.cnet.com/interview/story/0,2000050154,20096000-2,00.htm

[6] http://gree.jp/

[7] The excerpts of the conversation with Mr. Tanaka were drawn from a 1-hour semi-structured interview with him conducted in September 16, 2005 at his office in Tokyo, Japan.

both technological and market innovations (see Figure 4). This duality of roles of beta versions is a totally unprecedented notion for software development and would be a critical factor affecting the quality of service, relational structure among stakeholders, and hence final profits of the software business on the SaaS model.

As seen in the above-discussed cases, recent competition in service-oriented software is unfolding in rapidly changing environments. The SaaS model of software development and operation will considerably transform the existing frameworks and approaches for software development. In order to cope with this rising reality, software development of today must deal with various new development practices from a strategic management perspective.

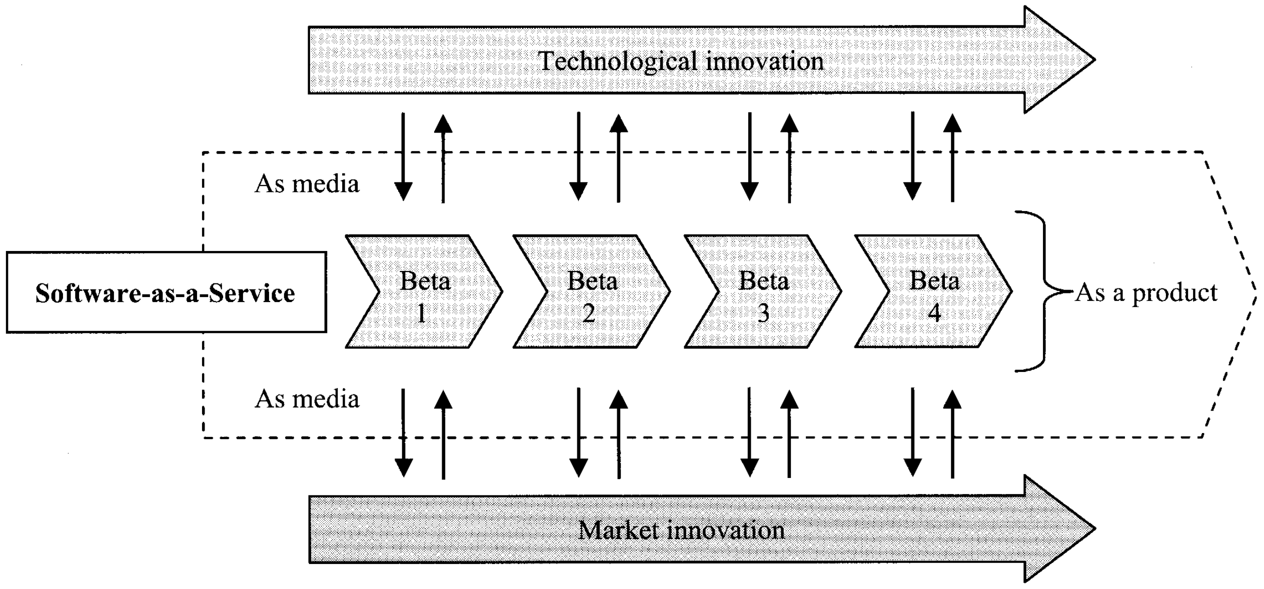

Figure 4: Dual roles of beta versions in the SaaS model

VII. Conclusion

The main objective of this paper was to explore today's service-oriented software development from a strategic management perspective. In summary, faced with increasingly dynamic and turbulent environments, software development practices must be managed through a constant 'dialogue' to technological and market innovations. Based on a review of the existing strategic management frameworks, Eisenhardt's framework of 'Strategy as Simple Rules' was discussed in terms of applicability to software development especially on the SaaS model. And with a brief case study of three online community services in Japan, the framework was proved to be well fit to SaaS development in rapidly changing environments. Moreover, the analysis of the cases suggested that beta versions could play dual roles in SaaS development, as a product and media, and that the duality would be a critical factor for strategy making in the service-oriented software business.

There are some limitations in the discussions of this paper. Most of the discussions unfolding in the paper are still hypothetical and clearly need empirical validation with both quantitative and qualitative methods. Furthermore, the paper only addresses SaaS development as a case of today's software development. Actual software development widely varies in its scale and settings. More detailed categorization of software development is clearly needed in future research.

REFERENCES

Barney, J.B. (2002). *Gaining and Sustaining Competitive Advantage.* (2nd edition) Pearson Education, New Jersey.

Cockburn, A. (2001). *Agile Software Development.* Addison-Wesley, Reading, MA.

Cusumano, M.A. (2004). *The Business of Software: What Every Manager, Programmer, and Entrepreneur Must Know to Thrive and Survive in Good Times and Bad.* Free Press, New York, NY.

Cusumano, M.A. and R.W. Selby (1995). *Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets and Manages People.* Free Press, New York, NY.

Eisenhardt, K.M. (1989). Making Fast Strategic Decisions in High-Velocity Environments. *Academy of Management Journal.* Vol.32, No.3, pp. 543-576.

Eisenhardt, K.M. and S.L. Brown (1998). Time Pacing: Competing in Markets That Won't Stand Still. *Harvard Business Review.* Vol.76, No.2, pp. 59-69.

Eisenhardt, K.M. and D.N. Sull (2001). Strategy as Simple Rules. *Harvard Business Review.* Vol.79, No.1 (January-February), pp. 107-116.

Eisenhardt, K.M. and B.N. Tabrizi (1995). Accelerating Adaptive Processes: Product innovation in the Global Computer Industry. *Administrative Science Quarterly.* Vol.40., No.1, pp. 84-110.

Heineman, G.T. and W.T. Councill eds. (2001). *Component-Based Software Engineering.* Addison Wesley, Reading, MA.

Highsmith, J.A. (1999). *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems.* Dorset House Publishing, New York, NY.

IDC (2005). *Worldwide and U.S. Software as a Service 2005-2009 Forecast and Analysis: Adoption for the Alternative Delivery Model Continues.* Doc #33120.

Kakihara, M. (2005). Software Development and the 'Simple-Rule' Strategy. (In Japanese) *Shogaku Ronkyu (Journal of Business Administration, Kwansei Gakuin University).* Vol.53, No.3, pp. 75-96.

MacCormack, A.D. (2001). Product-Development Practices That Work: How Internet Companies Build Software. *Sloan Management Review.* Vol.42, No.2, pp. 75-84.

McConnell, S. (2004). *Code Complete.* (2nd edition) Microsoft Press, Redmond, WA.

METI (2004). *A Report of the Field Study on Embedded Software Industry in 2004.* (In Japanese) Ministry of Economy, Trade, and Industry, Japan, Tokyo.

Mintzberg, H., B. Ahlstrand and J. Lampel (1998). *Strategy Safari: A Guided Tour Through the Wilds of Strategic Management.* Free Press, New York, NY.

Porter, M.E. (1980). *Competitive Strategy: Techniques for Analyzing Industries and Competitors.* Free Press, New York, NY.

Porter, M.E. (1985). *Competitive Advantage: Creating and Sustaining Superior Performance.* Free Press, New York, NY.

Pressman, R.S. (2004). *Software Engineering: A Practitioner's Approach.* McGraw-Hill, New York, NY.

Rust, R.T. and P.K. Kannan (2003). E-Service: A New Paradigm for Business in the Electronic Environment. *Communications of the ACM.* Vol.46, No.6, pp. 37-42.

Tamai, M. (1993). Current Practices in Software Processes for System Planning and Requirements Analysis. *Information and Software Technology.* Vol.35, No.6-7, pp. 339-344.