Endless Development is the Best Quality Assurance: The Case of “Hidemaru Mail”

Fumihiko IKUINE^{a)} and Hideki FUJITA^{b)}

Abstract: It is difficult to identify easily adaptable software in its completed form. In its nature, the definition of “complete” is unclear. As a result, if one expects perfection in software, software products will never be released; the only alternative is to release software if it runs, and then gradually refine it over time. This results in the implicit conventional wisdom that the “continuous development” of the software is the best quality assurance. This paper looks at the highly capable Windows e-mail software called “Hidemaru Mail.” With current progress in IT, it is possible that the concept of software quality assurance can be applied to a broader array of products and services.

Keywords: product development, software engineering, development project, product assurance myth, Hidemaru Mail

^{a)} Faculty of Engineering, Information and Systems Division of Policy and Planning Sciences, University of Tsukuba, fikuine@sk.tsukuba.ac.jp

^{b)} Undergraduate School of Business Administration, Toyo University, fujita-h@toyo.jp
A part of this paper was originally published as Ikuine and Fujita (2003) in Japanese.

Introduction

There is an implicit assumption that the development activity is a project and there is an “end” in theories to date. For example, Urban and Hauser’s text on the product development (Urban & Hauser, 1980, pp. 61–76) states that projects begin with the creation of a concept and end with the release of a product into the market (Urban & Hauser, 1980, p. 66, Figure 4.1). In Clark and Fujimoto (1991), a typical study on product development, a framework for analysis of actual activities (Clark & Fujimoto, 1991, p. 23, Figure 2.3, p. 27, Figure 2.4) shows that development activities seem to end.¹ Within inter-project continuity studies (Aoshima, 1997; Aoshima & Nobeoka, 1997; Nobeoka, 1996), continuity is flagged as an issue because of the assumption that a particular development will end.

However, in contrast with the development activity of hardware, that of software appears to have no “end”, for example, the birth of the IBM System/360 in the 1960s. Even though this system was released to its customers, the developers continued to fix software errors (or bugs) for several years after (Brooks, 1975/1995). In other words, the developers guaranteed the software’s quality assurance by continuous (endless) development right from its inception.

Of course, this concept of quality assurance is as par for the course today. Microsoft fixes bugs via monthly updates to Microsoft Windows. Social Networking Services (SNS) and other web services modify features and specifications within days, hours, or even minutes. Online software² distributed over the Internet has similar

¹ Kuwashima (2003) and Yasumoto (2006) are typical studies that analyze product development activities by turning projects into units.

² Software developed for online distribution (via the Internet) is referred to “online software” herein. Online software is not only known as free software, freeware or shareware, but many internet games and open source software (OSS) currently available in Japan are online software. Raymond’s (1997) pioneering research on online software primarily

characteristics, and many other software applications and services have frequent upgrades.

Despite this fact, existing research on enterprise information systems, operating systems on PC (Cusumano, 1991; Cusumano & Selby, 1995; Cusumano & Smith, 1993), and software development activity in the Internet era (Cusumano & Yoffie, 1998; MacCormack & Iansiti, 1997) states that software development activity has an end. It begins with a requirements definition and ends with software moving into phases of system operation and system maintenance.

However, in reality this is not the case. The Windows high functional e-mail application “Hidemaru Mail” of our case study has its origins in a separate e-mail application called NetMail. Hideo Saito, the developer of the “Hidemaru Editor” application for Windows, received a request to integrate the editor as a module into NetMail in the beginning. Later, a NetMail user group suspected that NetMail development was being abandoned, subsequently asking Hideo Saito to develop an e-mail application based on Hidemaru Editor. This shows that the user group was seeking quality assurance by asking Saito to continue development, even though at their expense. The responsibility of the development and distribution of Hidemaru Mail and the closely associated Hidemaru Editor was given to Saito Kikaku (Saito Enterprises), a limited company set up by Saito himself. This exemplifies the “continuous development” of the software, an implicit conventional wisdom for the best quality assurance. As a result, Saito continues to develop even though more than a decade

focused on OSS. Other studies include von Krogh, Spaeth, and Lakhani (2003); von Krogh and von Hippel (2003); and West (2003). However, these studies show tremendous affinity with various user innovation study groups (Jeppesen, 2005; Jeppesen & Frederiksen, 2006; Lüthje, 2004) beginning with von Hippel (1988). Thus, there is a strong tendency to focus on the formation and operation of communities, including users. On the other hand, there are not enough studies on software development activities.

has passed since the initial release.

Case Description: The Development of “Hidemaru Mail”³

In this paper, we used a case study method based on interviews with Saito and supplemented secondary materials, such as Hidemaru Mail release notes and information from the Internet.

Road to establishing Saito Kikaku

Hideo Saito was a self-taught programmer in 1980 and began developing software. After attending the National College of Technology, he found employment in a local subsidiary of a large IT corporation. However, he found no opportunities for programming and began independently developing software outside of work.⁴

Saito saw a business opportunity with the release of Windows 3.0 and DOS/V machines in 1990. He began to port the editor and communications software he had already written for other operating systems to a Windows 3.0 environment. Then he established his company, Saito Kikaku and released his software as shareware. In 1992 and 1993, he released the communications software HideTerm and Hidemaru Editor. Hidemaru Editor in particular gained a large user base, with about 150,000 paid subscribers in 2002, ten years after the initial release.⁵

Hidemaru Mail overview

Hidemaru Mail has 117,582 lines of code of program itself, excluding the Hidemaru Editor code it contains.⁶ When including the

³ The complete case study for Hidemaru Mail (in Japanese) can be found on http://www.gbrc.jp/onlinesoftware/case_Tsurukame.pdf

⁴ Saito noted that when he distributed his software to his colleagues, they were curious and enjoyed it.

⁵ This figure was given during an interview on October 1, 2002.

⁶ This number includes only line of code for *.h and *.cpp files, and does not

91,795 lines of code in Hidemaru Editor, the total lines of code in Hidemaru Mail become 209,377.⁷

Users must write macros to add their features to Hidemaru Mail because Saito Kikaku and Saito do not provide a kit for developers, a standard development kit (SDK), or a software interface. However, this does not mean that Hidemaru Mail limits the addition of unique functionality. In fact, Saito as well as users have developed and provided many macros for Hidemaru Mail and the built-in Hidemaru Editor.

History of Hidemaru Mail development

Saito began developing Hidemaru Mail after receiving a proposal from an Internet service provider named Xaxon. Xaxon asked Saito if they could include Hidemaru Editor as a module within their mail application “NetMail.” Saito gave his consent and a version of NetMail that included Hidemaru Editor, which went on sale as shareware in March 1998.⁸

However, because Internet Explorer 4 (on sale in September 1997) and Windows 98 (on sale in July 1998)⁹ included Outlook Express, many PC users viewed Outlook Express as the standard mail application in the Windows environment. In addition, with Becky! Internet Mail,¹⁰ AL-Mail,¹¹ and other Japanese mail applications expanding their market share, Xaxon unceremoniously stopped development of NetMail.¹² A user group that had come to distrust

include the installer and bundled software files.

⁷ Numbers effective December 13, 2002.

⁸ The price of the downloaded version was 4,500 yen, compared with the price of 12,800 yen for the package version (although the first 50,000 shipped were priced at 9,800 yen).

⁹ The English version was released in June.

¹⁰ Shareware developed by Tomohiro Norimatsu released for sale in 1996.

¹¹ Shareware developed by Tadashi Nakamura released in 1997 and development discontinued in 2006.

¹² Xaxon ceased new user registration in May 2000 and support in July of

Xaxon's attitude contacted Saito, who had been providing Hidemaru Editor. They asked Saito if he could develop a mail application that works based on Hidemaru Editor. In response to these requests, Saito began coding from scratch in April 2000, releasing Hidemaru Mail in August of that same year.

Software upgrades

Development of new versions of Hidemaru Mail began with users' requests and reports. Users would post requests for new functionalities based on features they noticed during usage and reported software bugs to the "Hidemaru Mail Support Room"¹³ or other related sites.

Among new requests for functionality are those which can be handled by macros. These are taken care of by macro development and new macro variables,¹⁴ and all other requests that cannot be dealt with by macros are taken care of by application upgrades of program itself. Responding to requests using macros is prioritized to avoid the risk of new bugs in the Hidemaru Mail application through upgrades. Undoubtedly, there are functions that cannot be dealt with by either macros development or upgrades. In these cases, Saito sometimes rejects requests to add functionality and explains to users that the particular functionality cannot be added in Hidemaru Mail.

Hidemaru Mail sites sometimes report crashes, freezes, and other abnormal phenomena. Hidemaru Mail is closed source software and such a program cannot be tested by user tools, which means that Saito must debug the program himself. However, because user environments widely vary among users, sometimes Saito cannot

that year.

¹³ <http://www.maruo.co.jp/hidesoft/8/index.html>

¹⁴ The addition of macro variables is done via libraries rather than the Hidemaru Mail application itself, though the overall Hidemaru Mail package is released as a new version.

pinpoint the source of a bug. In these situations, Saito provides a customized version of the software modified to output a dump file to certain users to debug (this fact was discovered in Hidemaru Mail's revised version history).¹⁵

Saito releases new versions of the software that include new features and fixes bugs without using performance testing or debugging. He accepts further requests for new features or reports of unresolved operation problems on his website, and responds to them in the next version of the software. Saito calls this a “constant open beta test.” Among newly refined versions of the software, he has a mixture of new features only, fixed bug only, or both.

Hidemaru Mail thus has an extraordinarily high frequency of upgrades (Table 1). In the 12 years, since the release of version .0 in August 2000, the frequency of upgrades was once every four days and that of resolved issues¹⁶ was approximately eight per upgrade.

Table 1. Version up history of Hidemaru Mail

	Solved problems a day	Development days a version	Solved problems a vesion	Number of versions	Amount of Days	Amount of problems
Beta	5.188	2.659	13.793	82	218	1131
Ver.1	3.050	4.673	14.255	98	458	1397
Ver.2	2.297	3.959	9.093	97	384	882
Ver.3	1.927	6.025	11.613	80	482	929
Ver.4	1.629	4.395	7.160	256	1125	1833
Ver.5	1.376	3.720	5.119	436	1622	2232
Total	1.959	4.089	8.011	1049	4289	8404

Note: Aggregated in May 22, 2012 by authors.

Source: From the website <http://hide.maruo.co.jp/software/tkhist.html>

¹⁵ <http://hide.maruo.co.jp/software/tkhist.html>

¹⁶ The term “issues” used herein refers to new features, specification changes, and bugs needing to be fixed.

For the version 5, released in December 2007, the frequency of upgrades was 3.7 days and that of resolved issues was five per upgrade.

Contact with users and its advantages

Saito Kikaku provides support for software users via a “support forum.” The support forum has “conference rooms” as subcategories that acted as bulletin boards. There are four Hidemaru Mail conference rooms divided by purpose. A user must register with the forum prior to posting on the bulletin boards, although viewing is open to anyone.

Postings on the Hidemaru Mail Support conference room for general users vary from questions on using Hidemaru Mail and additional feature requests to application operation reports. Saito generally answers questions and sometimes users offer advice. On the other hand, conference rooms for heavy users¹⁷ are divided by bug reports, feature requests, and macros available to the public. These three “conference rooms” often delve into detailed discussions on features and specifications.

Discussion

The request from Xaxon was a catalyst for Saito Kikaku’s venture. Moreover, Saito became involved in the development of an e-mail application. However, user groups’ concerns about the future of Xaxon’s Netmail roused Saito to continue the development of Hidemaru Mail.

Netmail’s future looked bleak, but users had several options such as Outlook Express included in Windows 98, the already released Becky! Internet Mail, or AL-Mail.

¹⁷ <http://www.maruo.co.jp/hidesoft/>

Despite these options, many NetMail users asked Saito, who until then was providing an editor module for NetMail, to develop Hidemaru Mail. Many requests are believed to stem from the fact that Hidemaru Editor, released in 1993, had gained many users and that Saito had continued development. Saito's conscientious workmanship in his development of Hidemaru Editor raised users' expectations.

In other words, Hidemaru Editor had developed a track record because Saito was a superior developer and continually developed the application while considering each user request and report as critical. Irrespective of the superior e-mail application he developed, if he quit development, the result would have been the same as that of NetMail. Thus, we see that user groups believed Hidemaru Mail was "likely" to have ongoing development and requested Saito to develop the application.¹⁸

Saito kept up to these expectations and continued responding to bug reports and additional feature requests by these user groups. Figure 1 shows how Saito and users groups circulated information—bug reports and new feature requests—over the course of more than ten years. Saito's continual engagement without "end" established user groups' belief in "continual development in the future." This then became the best quality assurance of Hidemaru Mail.

In contrast, it goes without saying that NetMail, along with the AL-Mail, released around the same time, ceased this information circulation at some point. As a result, the software developers failed to establish this belief and quality assurance as Hidemaru Mail did.¹⁹

¹⁸ See Kosuge (2007) for more information on the relationship between users and software providers and developers.

¹⁹ Becky! Internet Mail may be an application in which the establishment of the belief and the best quality assurance were realized in a manner similar to Hidemaru Mail.

Figure 1. Development cycle of Hidemaru Mail

Note: —→ activity;→ dependency

Source: Originally by authors.

Conclusion

Unlike tangible hardware products, software is easily adaptable. It is difficult to define the complete state of a software because it is unclear at what point the software may actually be finished. As a result, if perfection is expected, the software will never be released. Software is released if it runs and then refined gradually over time. This paper concluded that there is an assumption for software products: the “continuous development” of the software is the implicit conventional wisdom for the best quality assurance. Not only researchers but also practitioners are held captive by the belief that development activity has a defined end. This is probably because once a hardware product is completed and handed over to users,

developers can no longer do any work on the product. Another reason is that it is taken as par for the course that if they are, corporations will provide quality assurance on their products, repairing them and solving defects as necessary.

However, in the case of software, these facts and par for the course do not apply. It is possible for developers to continue working on software even when that software is in the hands of users. As can be seen in the case of Hidemaru Mail, whether developers continue working on software depends on the developer's responses and desire.²⁰

Further, distinctions between software and hardware as well as corporate perspectives on quality assurance are changing. A hardware product's firmware can be changed even after the product is released into the market. Perspective on how much quality assurance a corporation should provide is no longer as certain as it once was.

For this reason in particular, we must reconsider the idea that development activities have an end. Researchers and practitioners alike cannot ignore the phenomenon of the IBM System/360 released in the 1960s. The development activities we see do not have an end because it is possible that the continuation of development activity lead to the best quality assurance.

However, continuing development activities is cost incurring. In addition, there is the risk of competing with one's own products (cannibalization). Thus, careful consideration must be given, even after a product is released, to whether the assurance of "continual development in the future" will be given to users. Companies may then put together product and development strategies based on these considerations.

²⁰ Fujita and Ikuine (2013) discuss Saito's desire (or motivations) as a developer.

References

- Aoshima, Y. (1997). Shinseihin kaihatsu kenkyu no shiten [Perspectives on new product development]. *Business Review*, 45, 161–179 (in Japanese).
- Aoshima, Y., & Nobeoka, K. (1997). Purojekuto chishiki no manejimento [Management of project knowledge in new product development]. *Soshiki Kagaku*, 31, 20–36 (in Japanese).
- Brooks, F. P., Jr. (1975/1995). *The mythical man-month: Essays on software engineering* (Anniversary ed.). Boston, MA: Addison-Wesley.
- Clark, K. B., & Fujimoto, T. (1991). *Product development performance*. Boston, MA: Harvard Business School Press.
- Cusumano, M. A. (1991). *Japan's software factories: A challenge to U.S. management*. New York, NY: Oxford University Press.
- Cusumano, M. A., & Selby, R. W. (1995). *Microsoft secrets: How the world's most powerful software company creates technology, shapes markets, and manage people*. New York, NY: Free Press.
- Cusumano, M. A., & Smith, S. A. (1993). Beyond the waterfall: A comparison of “classic” versus PC software development. (Working Paper, 3607-93). Cambridge, MA: Alfred P. Sloan School of Management, Massachusetts Institute of Technology.
- Cusumano, M. A., & Yoffie, D. B. (1998). *Competing on Internet time: Lessons from netscape and its battle with Microsoft*. New York, NY: Free Press.
- Fujita, H., & Ikuine, F. (2013). Free and open source are not necessary conditions of successful development: The case of “Hidemaru-Mail.” *Annals of Business Administrative Science*, 12, 151–166. doi: 10.7880/abas.12.151
- Ikuine, F., & Fujita, H. (2003). Onrain sofuto no kaihatsu sutairu [Online software development style]. *Akamon Management Review*, 2(5), 231–238 (in Japanese).
- Jeppesen, L. (2005). User toolkits for innovation: Consumers support each other. *Journal of Product Innovation Management*, 22, 347–362.
- Jeppesen, L. B., & Frederiksen, L. (2006). Why do users contribute to firm-hosted user communities? The case of computer-controlled

- music instruments. *Organization Science*, 17, 45–63.
- Kosuge, R. (2007). Internal effects of customer contact within service organizations: Implications for developing a customer-oriented organizational culture. *Annals of Business Administrative Science*, 6, 1–13. doi: 10.7880/abas.6.1
- Kuwashima, K. (2003). Organizational capability and competitive advantage in pharmaceutical product development. *Annals of Business Administrative Science*, 2, 21–27. doi: 10.7880/abas.2.21
- Lüthje, C. (2004). Characteristics of innovating users in a consumer goods fields: An empirical study of sport-related product consumers. *Technovation*, 24, 683–695.
- MacCormack, A., & Iansiti, M. (1997). Developing products on Internet time. *Harvard Business Review*, 75(5), 108–117.
- Nobeoka, K. (1996). *Maruchi purojekuto senryaku* [Multi-project strategy: Post-lean product development management]. Tokyo, Japan: Yuhikaku (in Japanese).
- Raymond, E. (1997). *The cathedral and the bazaar*. Retrieved from <http://www.catb.org/~esr/writings/cathedral-bazaar/>
- Urban, G. L., & Hauser, J. R. (1980). *Design and marketing of new products*. Englewood Cliffs, NJ: Prentice-Hall.
- von Hippel, E. A. (1988). *The source of innovation*. Oxford, UK: Oxford University Press.
- von Krogh, G., Spaeth, S., & Lakhani, R. (2003). Community, joining, and specialization in open source software innovation: A case study. *Research Policy*, 32, 1217–1241.
- von Krogh, G., & von Hippel, E. A. (2003). Special issue on open source software development. *Research Policy*, 32, 1149–1157.
- West, J. (2003). How open is open enough?: Melding proprietary and open source platform strategies. *Research Policy*, 32, 1259–1285.
- Yasumoto, M. (2006). Reconsidering novel technology introduction strategies: Impacts of technology, design, and market attributes on 118 Japanese product development projects. *Annals of Business Administrative Science*, 5, 67–97. doi: 10.7880/abas.5.67