How to Avoid Fork: The Guardians of Denshin 8 Go

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

Abstract: In terms of software, “continuous development” of the software is the best quality assurance. Continuous development requires either the original developer to continue development, or the next generation of developers to take over the development. It has been noted that when the original developer has used “open source,” a development paradigm in which the source code is kept open to all, highly motivated and competent developers will participate in development. This makes it easier for a project to survive. However, at the same time, when the source code is opened to a broad audience a fork in the source code tends to occur. When this happens, those with a high level of competence and motivation abandon the development and go their separate ways. In such a situation, it is difficult for a project to survive. In the case of Denshin 8 go, a guardian for the source code was appointed, and the original developer and the guardians avoided this dilemma. Richard Stallman of the GNU project and Linus Torvalds of Linux project act as legitimate guardians to avoid fork.

Keywords: software development, open source, fork, buried source code, legitimate guardian, Denshin 8 go

^{a)}Division of Policy and Planning Sciences, Faculty of Engineering, Information and Systems, University of Tsukuba, 1-1-1 Tennodai, Tsukuba-shi, Ibaraki, Japan, fikuine@sk.tsukuba.ac.jp

^{b)}Faculty of Business Administration, Toyo University, 5-28-20 Hakusan, Bunkyo-ku, Tokyo, Japan, fujita-h@toyo.jp

A part of this paper was originally published as Fujita and Ikuine (2005) in Japanese and presented at the ABAS Conference 2014 Spring (Ikuine & Fujita, 2014).

Introduction

For software, “continuous development” of the software is the best quality assurance (Ikuine & Fujita, 2013). Continuous development requires either the original developer to continue development or others to take over (Fujita & Ikuine, 2013b). When other developers take over, the original developer must share the source code with others. Open Source is a paradigm wherein the source code is shared and made broadly available to others. It has been noted that the project adopting open source can attract competent and highly motivated developers. Thus, some claim that developers can readily ensure the survival of software development projects when the source code is open.

However, opening the source code to a broad audience can cause heterogeneity in the source code, even when people continue the development based on the original source code, a phenomenon known as “fork (forking).” In the case of open source software (hereinafter, “OSS”), literally anyone can obtain the source code and modify it to develop new software and start up new projects. Therefore, forks often occur in the OSS project, and the impact of forking has been studied for very long (Dibona, Ockman, & Stone, 1999; Kogut & Metiu, 2001; Nyman & Lindman, 2013; Nyman, Mikkonen, Lindman, & Fougère, 2012; Raymond, 1997, 1999). Moreover, some studies have focused on how to maintain forked projects, under the assumption that forking will inevitably occur (Gamalieleson & Lundell, 2013; Ray & Kim, 2012; Ray, Wiley, Kim, 2012).

When forking does occur, competent and motivated developers start to abandon the project and go their separate ways, which may affect the survival of these projects. In other words, sharing the source code comes with a dilemma. If one loosens restrictions and shares the source code broadly, that project may survive; however, if

the project is forked, competent successors with strong skills and desires will end up working on their own, or perhaps inter-project competition may lead to vying for competent successors. On the other hand, while establishing strict conditions and sharing the source code only with a select few may make the source code easy to control and reduce the possibility of forking, it may become more difficult to find competent successors willing to participate under those conditions. This puts the survival of the project at risk. Ensuring the survival of a project requires that one avoid this dilemma of source code sharing.

Research on avoiding the fork has heretofore focused on developer motivation, licenses, and project governance (Ernst, Easterbrook, & Mylopoulos, 2010; Neville-Neil, 2011; Nyman & Mikkonen, 2011; Robles & Gonzalez-Barahona, 2012; Viseur, 2012). However, forking is a phenomenon not limited to OSS, and examples can be seen in Unix as well (Takahashi & Takamatsu, 2002, 2013). Although not a fully OSS project, we will consider the case of Denshin 8 go, which has shared its source code and has been continuously in development for more than 10 years with a successful generation change of developers while managing to avoid forking.

No matter how good the license, it is meaningless if people cannot honor it. We focus on the developers understanding the intent of the license as well as understand how competent successors are appointed as guardians to protect the license during the continuous development.

Case Description: The Development of “Denshin 8 Go”¹

In this paper, we opted to use a case research method. This case study is based on interviews with the founder of Denshin 8 go

¹ The complete case study for Denshin 8 go (in Japanese) can be found at http://www.gbrc.jp/onlinesoftware/case_Den8.pdf

(pronounced “denshin hachi-go”), Takamitsu Ishioka, as well as a group of developers that took over the development from him. We supplemented the interviews with secondary material, release notes, and information obtained from the internet, and combined to form our case study.

An overview of Denshin 8 go

Ishioka began development of Denshin 8 go, an e-mail application, in 1995. Its development has been carried on by volunteer organizations called “Den8club” (pronounced as “denpachi club”) and “Den8dev club” (“denpachi dev club”). The functionality and sophistication of the application are highly regarded, and it won the 5th Free Software Prize (FSP’96) sponsored by the Internet Association Japan.

Characteristic of the software are its extremely small memory footprint, despite having the necessary functions one would expect of a mail client.² In addition, it has neither mail editing functions nor mail viewing functions so that users can choose their favorite editors and viewers for themselves.

Both the Den8club and the Den8dev club that are currently developing Denshin 8 go have mailing lists stating that “anyone can join and leave at any time.” The mailing lists of these two clubs are independent, with the difference being the high level of development activity in the list sponsored by the Den8dev club. The Den8dev club primarily distributes unstable alpha versions of the software and sponsors technology information exchanges related to development.

The latest formal build binary of Denshin 8 go is version 32.1.7.3, which was released on May 7, 2014. Thus, the software has been continually developing for approximately 19 years since the original

² The size of the Denshin 8 go application has grown approximately 50,000 lines since the time the original developer, Ishioka, began development, but is still not more than 100,000 lines.

developer began development, and approximately 15 years since the current development structure was put in place.³

The original developer Ishioka

Ishioka was compelled to develop Denshin 8 go because he wanted “an easy to use e-mail application,” as well as out of a desire to “make a little money.” E-mail client applications available for Windows at the time all seemed difficult to use. In the spring of 1995, Ishioka felt he wanted to make a simple Windows-based e-mail application, and thus began development.

During development, Ishioka modeled his software on the MH (mail handler) e-mail application widely used in Unix environments. He worked on the project every day in the evenings and on weekends, and by the summer of 1995, the software had become usable. By fall of that year, Ishioka felt the software was stable, and he posted it on an Asahi-net software bulletin board.

Fueled by his motivation,⁴ Ishioka created new versions of the software several times per month. He produced code based on his personal desires and the reactions of users, and once the code was compiled, he quickly released the software. However, he felt that releasing software without debugging and testing was a problem, and he released both a release version as well as a developer’s (beta) version in the summer of 1996. With the release of Windows 95 and the increasing use of the internet from around 1997, users of Denshin 8 go began to increase.

The transition of the development

However, the frequency of new version releases gradually began to slow around 1998. Users sent many requests by the same pace, and

³ Refer to “Internet Mailer Denshin 8 go Official Site” <http://www.denshin8.jp/#toc>

⁴ See Fujita and Ikuine (2013b) regarding Ishioka’s motivation.

Ishioka himself tried to respond to these requests, but at one point, Ishioka began to feel that the software was “good enough” for his personal use. Around that same time, another e-mail application called “Becky! (Becky! Internet Mail)” was released; it quickly expanded its user base and Ishioka’s initial hopes of making some money with Denshin 8 go grew dim. Thus, Ishioka’s passion for developing the software began to decrease.

Consequently, Ishioka became unable to respond to the constant stream of user requests. While Ishioka would receive requests to improve Denshin 8 go, he kept the source code locked up and was unable to respond to those requests. Ishioka began feeling like this situation was a burden.

One day, in July 1999, Ishioka visited the Den8club website and was struck by the presence of so many passionate users and operators of the website. Ishioka then felt that it was wrong to keep the source code locked away with so many requests and passionate users of the software. He decided to allow access to the source code to others.

Ishioka contacted Kenichiro Nakamura, who was very active on the Den8club mailing list, and on July 13, 1999, Ishioka opened up the source code with certain conditions attached. Those conditions were 1) the provision of a distribution server and 2) not allowing multiple forks of Denshin 8 go.⁵

The rise of Den8club build

The Den8club required a substantial amount of preparation to take over development. First, the club had an internal discussion regarding securing a distribution server to fulfill Ishioka’s requirement of sharing the source code. These discussions had

⁵ These conditions are currently being met, with only the Den8dev club, Den8Club, and “Nobinobi Kyouiku Kenkyuukai” having the source code besides Ishioka himself.

already been taking place with members of what would become the Den8dev club, rather than with the general membership of the Den8club.

Once it appeared that the server discussion was on the right track, the club appointed an “official builder”⁶ to ensure that Denshin 8 go would not fork into multiple projects, another of Ishioka’s requirements. They decided that the official builder would be responsible for version control, and the role was then documented. This role encompassed source code control, build control, and site control. When Nakamura had an occasion to purchase relevant equipment, he also entered his name as a candidate for official builder. After gaining the trust of the group, Nakamura was assumed in the role on August 22, 1999. An official build binary (version 321.2b6) was released only to the Den8club on August 28, 1999.

However, soon after the Den8dev club took over the development, development did not go as expected. No one was processing update patches, and these patches were held up for almost a year, with none of these patches being reflected into the binary versions.

It was in these circumstances that Takahiro Fukui⁷ came forward as a candidate for the official builder role, with the intention to “give feedback to the users.” Fukui believed “having only patches meant that, while certain individuals with the ability to create personal

⁶ The “official builder” is a person who works on development of beta versions.

⁷ Fukui also participated in the Den8club in December 1996. However, his participation in the club was limited to that of a “ROM” (read-only member), and not a particularly active member. Further, he considered discontinuing his personal use of Denshin 8 go in 1998 because of lack of new releases and problems not being resolved. Despite that, Fukui decided to make a strong commitment to Denshin 8 go when the code was shared in 1999. Fukui had programming experience, and he agreed with the idea of open sourcing. He thus participated in the founding of the Den8dev club, becoming responsible for development after Ishioka opened up the access to the source code in July 1999 and discovering that anyone could participate in its development.

builds could use Denshin 8 go after improving it and removing bugs, many users could not use the maintained version, which was both unfair and absurd.” At the same time, he had a desire to “create a better Denshin 8 go usable by anyone, and used by all,” and thus submitted his candidacy for the role of official builder.⁸

Upon completing the prescribed procedures, Fukui became the official builder on June 12, 2000. Fukui’s policy at the time was to “take care of all the patches that were held up.” He did not touch Den8dev club’s first build (version 321.2b6-stable),⁹ which only provided bug fixes, and instead focused his energies on the next public release (v321.2a71), which improved functionality and fixed issues using all the patches that had been building up. The first one or two months were spent working almost exclusively on Denshin 8 go. Rapid releases of an alpha version¹⁰ on a weekly basis made the releases seem like a weekly magazine, “Weekly Den8.” In this manner, the beta version of v32.1.3.0 saw frequent releases internally within the Den8club between July and October 2000, and the official build binary of Denshin 8 go version 32.1.3.1 saw a general release on December 24, 2000.

The origin of Den8club, Den8dev club

Ishioka created a bulletin board in the early days of the Denshin 8 go development from the end of 1995 to the beginning of 1996, to avoid having to respond to the same questions that had previously been asked by users. At the same time, users had separately set up

⁸ In addition, some supported Nakamura, though it was obvious through the mailing list that the club was in a difficult position, and asked Nakamura to take on the role of updating patches while Fukui would create builds.

⁹ The developer’s version was released on March 15, 2000 and the final version on March 31, 2000.

¹⁰ Starting around this time, the practice of the Den8dev club to release a particularly unstable alpha version as an internal only release was established.

their own mailing list, which evolved into a forum for taking suggestions for Denshin 8 go requirements.

The Den8club began as a website created by Hiroshi Kawase in 1995 as a place to exchange information among users. In 1996, the site had about 40 participants, and from the outset compiled requests and submitted them to Ishioka. Later, the Den8club became the forum for communication between Ishioka and users, and Ishioka used the Den8club to distribute developer beta versions of the software. Since then, the Den8club has maintained its function as a forum to share information among users. Further, a separate mailing list, the Den8dev club, began just as the source code of Denshin 8 go was shared in July 1999.

Regarding the period when Ishioka handed off the source code to the Den8Dev club and Den8club, Yoshihisa Honda stated, “the Den8club was originally a gathering place for users, and there was some confusion when the source code was entrusted to the club.” At the time of the handover, there was friction between the existing Den8club and the newly formed Den8dev club, which was created to continue the development effort. Some people expressed dissatisfaction with the conditional nature of the source code release. These individuals left the Den8club.

There is no leader in the Denshin 8 go development effort. Nakamura terms this a “loose-ended” and “unstable” situation, and states “the development of Denshin 8 go is modeled on BSD.” For his part, Fukui notes, “Ishioka casts a large shadow, but the role of Nakamura, who has explained things passionately and worked on official builds, has been extraordinary.”

What is the “Denshin 8 go”

For Nakamura, Denshin 8 go is an e-mail application with additional elements fully compliant with RFC (Request for Comments). Thus, he takes care that the Denshin 8 go itself and the

application plugins are segmented. In addition, though the application has undergone extensive internal changes with the additional versions, the look and feel remains unchanged from the original version. In other words, Nakamura has striven to create official builds that are fully RFC compliant, while keeping external specifications intact¹¹ and maintaining the look and feel, and improving the internal behavior and functionality as much as possible.¹²

On the other hand, Fukui's concerns regarding official build creation are more about how others use the software, so the issue of whether the software has fatal bugs is crucial. Fukui is extremely "mechanical" in building the binary, and is careful that his originality as a programmer is not too evident in Denshin 8 go.

The meaning of the Denshin 8 go way

Denshin 8 go has been a free software since its inception, and it currently is not only free but also conditionally open sourced. Some are critical of these conditions, which they regard as being "too strict." However, Nakamura feels they are "reasonable" given the copyright law, the cultural and historical backgrounds in Japan.

In addition, when both software and source code are open, software versions can fork, causing the software with the same name and roots to lose compatibility between versions. Such a problem has happened in the past. Forking likely occurs when there is no leadership, or when the leadership is weak. However, in the case of

¹¹ Simply put, "links between Denshin 8 go and external software should function smoothly."

¹² Some people are satisfied with the direction Nakamura has chosen, while some are not. This dissatisfaction centers on the belief that "nothing with Denshin 8 go has changed." However, Nakamura states, "having major version releases where it is obvious that 'something changed' only causes problems where people are left wondering what kind of software Denshin 8 go actually is. Complete rewrites are not what Denshin 8 go is about."

Denshin 8 go, the original developer, Ishioka, handed off the source code and transferred development of the software under the condition that the principal developers and development targets would be combined. As a result, the project has not forked.

Discussion

In the case of Denshin 8 go, Ishioka, the original developer, was unable to continue development, and therefore entrusted the development of Denshin 8 go to competent and motivated successors. This change in developers occurred through making the source code shared with conditions.

There was a possibility of a forking when sharing the Denshin 8 go source code. Should that have happened, highly competent successors would split off, putting the survival of the project at risk. In addition, forking can cause multiple, similar versions of the software to appear, and it was possible that the Denshin 8 go created by Ishioka would have morphed into something similar yet unrecognizable.

When several versions of software appear in the process of development, both developers and users are inconvenienced. Users must determine which software is the same as before a fork, as well as understand which software can use existing data. The case of LibreOffice, which Gamalieleson and Lundell (2013) researched, and the case of BSD, of which multiple versions were developed, are examples of the impact of a fork on users. The fact that Unix did not experience additional forks beyond the first may have ensured the success of Linux. In light of that, one must think of ways to avoid fork.

In the case of Denshin 8 go, Ishioka, Nakamura, and the Den8dev club members were aware of the dangers of fork. That is why Ishioka suggested conditions and Nakamura and other Den8dev club

members adhered strictly to those conditions. In other words, in the case of Denshin 8 go, Ishioka was able to find competent successors in the form of Nakamura and others, who were willing to accept his conditions and take on development. These developers became the guardians of the software. Sharing the source code only with a select few of the Den8Club helped avoid forks and ensured the survival of the project.

Through this case study, can it be said that the dilemma of source code sharing was avoided by developer motivation, licenses, and project governance? Were Nakamura and other Den8dev club members motivated by suitable conditions regarding source code sharing? Did the Den8dev club members conform to these conditions because of the good governance structure inherent in Den8dev club?

Even with good governance structures and well thought out licensing, if successors cannot agree, they will lose desire and passion, and they will not take over a project. Rather, successors motivate themselves by creating their own licenses and governance structure based on conditions proposed by the original developer. In that case, it is critical to appoint a guardian from among competent successors, who understands the conditions for sharing the source code.

In other words, how best to govern a project is not a critical factor in deciding whether to share the source code via an open source model (Fujita & Ikuine, 2013a). More important is having a legitimate guardian of the source code, someone who has both the will and the competence. Richard Stallman of the GNU project and Linus Torvalds with Linux are just such examples. These projects have different licenses and different governance structures, but they do have in common the existence of legitimate guardians.

Conclusion

Sharing a source code brings with it a dilemma. Unconditional source code sharing, or opening up the source code with very loose restrictions increases the likelihood of attracting motivated and competent successors to a development project. However, at the same time, it increases the risks of forking. On the other hand, when a source code is shared with restrictive conditions, the risk of forking decreases. However, the likelihood of attracting competent successors who will agree to the conditions also diminishes. In other words, it is difficult to strike a balance between avoiding fork and attracting competent successors. Accordingly, how one avoids this dilemma affects project continuity and determines the success of a project, as well as whether that project can provide the best quality assurances.

This dilemma of source code sharing cannot be avoided simply by adopting open source. No matter how good the license or how skillful the project governance, it is meaningless without trusting the people who will do the work. Rather, appointing the legitimate guardians of the source code and then entrusting a project to that guardians is critical.

Thus, the issue then becomes finding competent successors and appointing the legitimate guardians of the source code. These guardians can only be selected prior to sharing the source code and by assessing candidates. In the example of Denshin 8 go, when Ishioka was still developing it, he discovered Nakamura, who had proven himself as someone with whom he could entrust the project. Nakamura accepted the conditions for opening up the source code suggested by Ishioka, became the official builder, and adhered to the agreed-upon conditions. Nakamura's actions attracted the contributions of the other Den8dev club members, including Fukui. Nakamura, Fukui, and other members of the club became guardians

of the source code, and have continued development for more than 10 years, ensuring the essence of Denshin 8 go.

References

- Dibona, C., Ockman, S., & Stone, M. (1999). *Open sources: Voices from the open source revolution*. Sebastopol, CA: O'Reilly.
- Ernst, N. A., Easterbrook, S., & Mylopoulos, J. (2010). Code forking in open-source software: A requirements perspective. *CoRR*, arXiv:1004.2889
- Fujita, H., & Ikuine, F. (2005). Datsupakkejika shita sofutouea no kaihatsu: Sofutouea kaihatsu niokeru yuza no shoshikika [The development of software free from package: Organizing users in software development]. *Akamon Management Review*, 4, 51–70 (in Japanese).
- Fujita, H., & Ikuine, F. (2013a). 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
- Fujita, H., & Ikuine, F. (2013b). Open source, a phenomenon of generation changes in software development: The case of Denshin 8 go. *Annals of Business Administrative Science*, 13, 1–15. doi: 10.7880/abas.13.1
- Gamalieleson, J., & Lundell, B. (2013). Sustainability of open source software communities beyond a fork: How and why has the LibreOffice project evolved? *Journal of Systems and Software*, 89, 128–145. doi: 10.1016/j.jss.2013.11.1077
- Ikuine, F., & Fujita, H. (2013). Endless development is the best quality assurance: The case of “Hidemaru-Mail.” *Annals of Business Administrative Science*, 12, 251–263. doi: 10.7880/abas.12.251
- Ikuine, F., & Fujita, H. (2014, May). *How to avoid fork: The case of Denshin 8 go*. Paper presented at ABAS Conference 2014 Spring, University of Tokyo, Japan.
- Kogut, B., & Metiu, A. (2001). Open-source software development and

- distributed innovation. *Oxford Review of Economic Policy*, 17, 248–264. doi: 10.1093/oxrep/17.2.248
- Neville-Neil, G. V. (2011). Kode vicious: Think before you fork. *Communication of the ACM*, 54, 34–35. doi: 10.1145/1953122.1953137
- Nyman, L., & Lindman, J. (2013). Code forking, governance, and sustainability in open source software. *Technology Innovation Management Review*, (January, 2013), 7–12.
- Nyman, L., & Mikkonen, T. (2011). To fork or not to fork: Fork motivations in SourceForge projects. *International Journal of Open Source Software and Processes*, 3(3), 1–9. doi:10.4018/jossp.2011070101
- Nyman, L., Mikkonen, T., Lindman, J., & Fougère, M. (2012). Perspective on code forking and sustainability in open source software. *Open Source Systems: Long-Term Sustainability*, 378, 274–279. doi: 10.1007/978-3-642-33442-9_21
- Ray, B., & Kim, M. (2012). A case study of cross-system porting in forked project. In *Proceedings of the 20th ACM SIGSOFT International Symposium on the Foundation of Software Engineering*, November 2012, Cary, CA. doi: 10.1145/2393596.2393659
- Ray, B., Wiley, C., & Kim, M. (2012). Repertoire: A cross-system porting analysis tool for forked software project. In *Proceedings of the 20th ACM SIGSOFT International Symposium on the Foundation of Software Engineering*, November 2012, Cary, CA. doi: 10.1145/2393596.2393603
- Raymond, E. S. (1997). *The cathedral and the bazaar*. Retrieved from <http://www.catb.org/~esr/writings/cathedral-bazaar/>
- Raymond, E. S. (1999). *The magic cauldron*. Retrieved from <http://www.catb.org/~esr/writings/cathedral-bazaar/magic-cauldron/>
- Robles, G., & Gonzalez-Barahona, M. (2012). A comprehensive study of software forks: Dates, reasons and outcomes. *Open Source Systems: Long-Term Sustainability*, 378, 1–14.
- Takahashi, N., & Takamatsu, T. (2002). Opun sosu senryaku no gokai: Naze Linux wa seikoushitanoka [Misleading about the open source strategy: Why Linux succeeded?]. *Akamon Management Review*, 1, 283–308 (in Japanese).

- Takahashi, N., & Takamatsu, T. (2013). UNIX license makes Linux the last major piece of the puzzle. *Annals of Business Administrative Science*, 12, 123–137. doi: 10.7880/abas.12.123
- Viseur, R. (2012). Fork impacts and motivations in free open source projects. *International Journal of Advanced Computer Science and Applications*, 3, 117–122.

Received May 30, 2014; accepted June 27, 2014