LETTER

Software Maintenance Evaluation of Agile Software Development Method Based on OpenStack

Yoji YAMATO\(^{(a)}\), Member, Shinichiro KATSURAGI\(^{†}\), Shinji NAGAO\(^{†}\), and Norihiro MIURA\(^{†}\), Nonmembers

SUMMARY We evaluated software maintenance of an open source cloud platform system we developed using an agile software development method. We previously reported on a rapid service launch using the agile software development method in spite of large-scale development. For this study, we analyzed inquiries and the defect removal efficiency of our recently developed software throughout one-year operation. We found that the defect removal efficiency of our recently developed software was 98%. This indicates that we could achieve sufficient quality in spite of large-scale agile development. In term of maintenance process, we could answer all inquiries within three business days and could conduct version-upgrade fast. Thus, we conclude that software maintenance of agile software development is not ineffective.

key words: agile software development method, software maintenance, defect removal efficiency, cloud computing, OpenStack, open source software

1. Introduction

Recently, many providers have started cloud services. For Infrastructure as a Service (IaaS), adoptions of open source software OpenStack\(^{[1]}\) are increasing. Regarding software development processes, the agile software development method\(^{[2]}\) (we shorten this to agile development) has spread widely. Agile development iterates short-term developments to adapt to external environmental changes.

It is generally said that agile development is not suitable for large-scale development involving more than 20 members in the team\(^{[3]}\). However, we adopted agile development for large-scale cloud platform (PF) development to quickly launch production services. Thanks to ideas of automatic regression techniques of Integration Tests, we could start our services relatively quickly\(^{[4]}\).

There have been many studies on evaluating agile development using the software reliability growth curve or development productivity (for example,\(^{[5]}\)). However, there is not much work on agile development evaluation including the software maintenance phase. Svensson and Host\(^{[6]}\) evaluates efforts and project velocity of agile development for an already delivered product but they do not evaluate quality such as the defect removal efficiency. Therefore, we analyzed the number of defects (such as software bug) and maintenance work throughout one-year operation, and we re-evaluated agile development including the maintenance phase.

2. Review of Developed System and Method

Our developed system in [4] is an IaaS cloud PF and mainly consists of OpenStack and a resource management server. Figure 1 shows an overview of our cloud PF.

OpenStack is an open source IaaS software and provides functions and REST APIs to control virtual resources. For example, Nova is a component to control virtual machines. A resource management server provides functions and business logic, which cover insufficient functions of OpenStack such as transaction management and effective deployment logic of virtual resources. Users or operators request virtual resource controls to OpenStack via the GUI/API of the resource management server. Functions of resource management server are described in [7].

There are more than 30 members in our development team and we developed the cloud PF system recursively with two-week iterations. We customized and fixed some bugs of OpenStack (version Folsom) by using Python. We newly developed a resource management server of which its GUI and API are implemented using Java, while the function that serves as intermediary between the resource management server and OpenStack is implemented using Phyton. The

---

\(^{(a)}\) E-mail: yamato.yoji@lab.ntt.co.jp

1 DOI: 10.1587/transinf.2015EDL8049
resource management code scales to several hundred kLines and the customized code of OpenStack scales to 10 kLines. Note that all codes of OpenStack Folsom are 550 kLines.

3. Software Maintenance Organization

Our operation company operates and sells a cloud service, and some developers including the authors maintain cloud PF software. There are less than ten maintenance members and each member has other roles. The maintenance team receives inquiries or defect reports from an operation company and answers them. The maintenance team also prepares a software version-upgrade that includes additional functions and many defect fixes.

Normal maintenance work involves a ticket system. When an operation company encounters a cloud PF problem that they cannot solve, the problem is reported to a maintenance desk via the ticket system. The reported ticket is copied to an internal Redmine ticket system and the event is analyzed first to determine whether the cause is from OpenStack, the resource management server, or others. Then, our maintenance team investigates the details of the cause, determines solutions such as a patch or operation procedure, and reports them to the operation company. The maintenance target is to provide responses within three business days.

4. Defect Status through Production Maintenance

We analyzed defect status of our cloud service for one year operation.

4.1 Ticket Inquiry Status

Figure 2 shows the inquiry status via a ticket system. The vertical axis shows the inquiry number/total inquiry number of each month and the horizontal axis shows elapsed months from service launch.

This figure indicates that the inquiry number increased from the 3rd month, was highest in the 4th month, then decreased but increased again after version-upgrade. The numbers of inquiries of OpenStack and resource management server were almost the same. While the code scale of OpenStack, which includes community codes and our customized codes, was larger than the resource management server code scale, many defects of OpenStack were already fixed because OpenStack is open source software. Therefore, the number of inquiries of OpenStack remained almost the same as that of the resource management server.

On the other hand, the highest inquiry number is in the 4th month. One reason was that we took 3-4 months to gather enough users, then users found many defects in the 4th month. Another reason was that virtual resource deployment logic was complex and took a time to discover defects. When we conducted version-upgrade in the 8th month, there were already a sufficient number of users, so the number of inquiries increased rapidly.

4.2 Classification of Ticket Inquiry and Defect Removal Efficiency

Figure 3 shows classifications of all inquiries starting from service launch until one year operation. The inquiries are not only defects but also problems of middleware (e.g. OpenSSL) vulnerability. Regarding OpenStack inquiries, design and environmental problems are in the majority. Regarding resource management server inquiries, specification and design problems are in the majority. We previously reported that the defects of specification and implementation problems were in the majority in agile development compared to waterfall development [4]. Figure 3 indicates that specification problems remained on the resource management server even in the operation phase.

Next, we calculated the defect removal efficiency of our cloud PF development. The defect removal efficiency is the [defect number discovered in the development phase] / ([de-
fect number discovered in the development phase] + [defect number discovered throughout one-year operation]) and is a metric of software process evaluation to show how many defects are discovered in the development phase.

We regard inquiries that are classified to specification, design, or implementation problems as defects throughout one-year operation. In our previous work [4], we have counted the number of defects in the development phase. Based on the number of defects found in the development phase [4] and the total number of defects throughout one-year operation, the defect removal efficiency of the resource management server is 98%. Jones analyzed 7,000 software projects in which the defect removal efficiency average of Japanese software is 93% and software development effort average is at least 98% of defect removal efficiency [8]. Therefore, we can conclude that the defect removal efficiency of our service (i.e., 98%) is sufficient and our agile development can keep sufficient quality regardless of large-scale development.

We cannot analyze the defect removal efficiency of OpenStack because the causes of almost all inquiries are in the community codes. We think we need to consider software quality evaluation methods for open source software itself or a system based on open source software.

5. Evaluation of Software Maintenance

We evaluate our one-year maintenance effectiveness in this section. The main maintenance work includes normal maintenance with a ticket system, software version-upgrade support, and community proposal of OpenStack fixes.

5.1 Normal Maintenance Work

Software maintenance in agile development is generally relatively ineffective because documents are not rich compared to waterfall development.

For a ticket investigation, we obtained logs, analyzed them to determine whether the resource management server or OpenStack that has caused the reported problem. To quickly answer tickets, we first checked OpenStack request results of the resource management server logs and isolated the cause. After our investigation, we prepared a software patch or reflected fixes to a version-upgrade package if codes fixes were needed. We answered all tickets within three business days throughout one-year maintenance. Vulnerability checks of middleware were urgent, so we answered them within one day. Responses within three business days are service levels of the NTT Open Source Software Center [9]. Therefore, we conclude that our response speeds were sufficient. Large-scale agile development is not ineffective for software maintenance because we can keep a sufficient maintenance level.

5.2 Version-Upgrade Support

Additional functions and defect fixes are introduced in version-upgrade. The version upgrades for resource management server and OpenStack were conducted eight months after service launch. The main additional functions are virtual resources orchestration and virtual resource fast restoration. These were also developed through agile development. Defect fixes included several patches of no emergent defects that were found through production operation. Note that emergent defects were quickly patched.

Because version-upgrade requires order stops, we did them during a midnight maintenance window. Developers, including the authors, prepared version-upgrade procedures. An operation company conducted production work after a rehearsal of test environments based on these procedures.

To reduce the number of specification misunderstandings problems, our operation company confirmed the behaviors of main functions in the early stage. Because we developed additional functions by agile development, we could provide our middle state software to our operation company one or two months before formal delivery date. Therefore, production version-upgrade was done two months after we provided procedures and a latest package. We found that we could introduce a new version fast enough.

5.3 Defect Reflection to OpenStack Community

We fixed OpenStack defects first, then the fixes were patched to a production system or reflected to a version-upgrade package. This is because the schedule of a community version defect fix depends on the community’s priority and we need to quickly fix fatal defects for our service. However, these fixes result in a gap between the community version and our customized version. Therefore, we proposed our fixes to a new community version of OpenStack in parallel with our original fixes.

However, major release of OpenStack is twice a year, and proposed fixes can be reflected to two versions later from the latest version. The gap between the community version and our customized version remains for about one year, which we think is a maintenance problem.

6. Conclusion

We evaluated the software maintenance of our cloud PF system developed using agile development throughout one-year operation. The highest number of enquiries occurred four month after service launch and those of OpenStack and resource management server were almost the same. Because the defect removal efficiency of the resource management server is 98%, we could achieve sufficient defect removal efficiency even though large scale agile development. Software maintenance in agile development is not ineffective because we could answer all tickets within three business days. We think the gap between the community version and our customized version is a maintenance problem because OpenStack defect fixes take times to reflect in the community version.
References