LETTER

Evaluation of Agile Software Development Method for Carrier Cloud Service Platform Development

Yoji YAMATO†, Naoko SHIGEMATSU†, Members, and Norihiro MIURA†, Nonmember

SUMMARY In this paper, we evaluate a method of agile software development for carrier Cloud service platform development. It is generally said that agile software development is suitable for small-scale development, but we adopt it for the development which has more than 30 members. We attempted to enable automatic regression tests for each iteration when we adopted agile software development, so that we could start our Cloud service sufficiently fast. We compared and evaluated software reliability growth curves, regression test efforts and bug causes with waterfall development.

key words: agile software development, cloud computing, OpenStack, automatic test

1. Introduction

In these days, agile software development method[1] has been spread widely which iterates short term developments to adapt external environment changes. For example, [2] is a work which evaluates software quality of agile software developments. However, agile software development is also called lightweight software development and is generally said that following areas are not suitable.

- Developments which members are more than 20.
- Geographically separated developers.
- Developments for mission critical systems.
- Company cultures based on instructions.

We have developed a Cloud service platform using open source software OpenStack[3] under above 4 conditions. Recently, many Cloud providers have started Cloud services and those developments are very fast. Therefore, we think we should develop Cloud services fast by adopting agile software development regardless of these 4 conditions. In this paper, we explain key ideas of agile Cloud service platform development and evaluate them.

2. Development System and Adopted Practices

2.1 Development System

A development system is an IaaS (Infrastructure as a Service) service platform and it consists of OpenStack and a resource management server. IaaS service provides hardware resources such as CPU and Disk via a network (for example, see the definition of the United States NIST[4]). Figure 1 shows a system outline of our Cloud service platform.

OpenStack is a project of open source IaaS software and it 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 logics which cover insufficient functions of OpenStack such as transaction management, physical resource management and effective deployment logic of virtual resources. Users or operators request to OpenStack via GUI/API of resource management server. Functions of resource management server are described in [5].

Development members are more than 30. We fixed some bugs and customized OpenStack (version Folsom) by Python. We newly developed a resource management server. Its GUI and API were implemented by Java (JDK1.6.0) and an intermediate function with OpenStack was implemented by Python.

2.2 Adopted Practices

There are some methods of agile software development such as eXtreme Programming (XP), Scrum and Feature Driven Development but we do not select one method. To com-
which calls all GUI and API functions at least once and also have a use case scenario which emulates user actual usages of multiple virtual resources creation, update and deletion. We re-used test cases and data of previous iteration IT tests for automatic regression tests. This idea reduced creation costs of automatic test cases and found degradations soon.

3.3 Frequent Coordination with Service Operators

We (developers) and operators proceeded specification discussions, development of Cloud platform and existing OSS/BSS coordination functions in parallel. This is because it takes more than one year to launch Cloud services if we precede these tasks sequentially. Minor specifications remained vague at the start of development, but those were discussed in parallel with development and reflected to a iteration in late stages. For operator team’s development of OSS/BSS coordination functions such as a billing system and an authentication system, we provided them a Cloud platform of middle iteration every two months.

4. Evaluation of This Agile Software Development

We could start our production services faster than normal waterfall development. In this section, we evaluate our agile software development comparing our previous Service Delivery Platform (SDP) development [7]. Except for waterfall development, the SDP development characteristics are similar to this development; more than 20 members in plural offices, a carrier service platform which needs a high reliability and a middle-scale development.

4.1 Software Reliability Growth Curve

Figure 3 (a) shows an IT/ST software reliability growth curve of this development and Fig. 3 (b) shows its of SDP development. The curve of Fig. 3 (a) is different from Fig. 3 (b) because found bugs are increased in proportional to development term and do not seem to be converged. Figure 3 (b) shows that most bugs are found by 3rd week of IT/ST and little bugs are found in 4th and 5th week. This is because waterfall development releases all functions in the beginning of IT/ST but agile software development releases functions partly in each iteration.

In the other side, Fig. 3 (c) shows an IT/ST software reliability growth curve of a certain function (Floating IP manage function) of certain iteration (2nd iteration) in this development. Regarding to one function, found bugs are almost converged about 2 weeks and little bugs are found later by a 48 hour continuous running test. Therefore, we should judge potential remained bugs not only total software reliability growth curve but also each iteration bug convergence.

4.2 Regression Test Efforts

Regression tests of 200 pages GUI and 60 APIs were automatically done in parallel with each iteration IT/ST. We reused about 5% of previous iteration IT test cases, test data

<table>
<thead>
<tr>
<th>Practice</th>
<th>Adopt or not</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small releases</td>
<td>☑</td>
<td></td>
</tr>
<tr>
<td>Metaphor</td>
<td>☑</td>
<td></td>
</tr>
<tr>
<td>Strike design</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td>☑</td>
<td></td>
</tr>
<tr>
<td>Refactoring</td>
<td>☑</td>
<td></td>
</tr>
<tr>
<td>Pair programming</td>
<td>☑</td>
<td>Each programmer makes source codes by himself/herself</td>
</tr>
<tr>
<td>Collective ownership</td>
<td>☑</td>
<td>We work more than 40 hour/week sometimes</td>
</tr>
<tr>
<td>Continuous Integration</td>
<td>☑</td>
<td></td>
</tr>
<tr>
<td>40-hour week</td>
<td>☑</td>
<td></td>
</tr>
<tr>
<td>On-site customer</td>
<td>☑</td>
<td>Programmers, testers and customers are in different offices</td>
</tr>
<tr>
<td>Coding standards</td>
<td>☑</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 2 Adopted XP practices of this development.
and automated regression test cases. When we automated one regression test, we needed about 3-5 times efforts of doing one IT test for Selenium GUI test and about 3 times efforts of it for Python API test. Because iterations became 10 times finally, automatic regression tests could reduce regression test efforts about 10-20% compared with that of manual tests. Therefore, we could pay more efforts for IT/ST/RT tests.

Figure 4 is a rough comparison of regression test efforts with automatic tests and manual tests. The vertical axis shows regression test efforts/each iteration IT efforts. Although we needed some efforts of creating automatic test cases for new functions of each iteration, we could reduce total efforts for many iterations because regression tests could be automatically done in night.

In SDP development, we did manual regression tests of 5% IT/ST test cases once in the last stage of IT/ST. Regression test efforts of agile software development are larger than those of waterfall development because we have many iterations. If we did manual regression tests in this develop-

Fig. 3  Software reliability growth curves of IT/ST bugs. (a) total system in this development, (b) total system in SDP development, (c) a certain function of certain iteration in this development.

Fig. 4  Comparison of regression test efforts.

Fig. 5  Ratios of bug causes. (a) this development, (b) SDP development.

ment, Fig. 4 indicates total regression test efforts need more than twice efforts of SDP development. Adopting automatic tests, we could keep regression efforts less than twice of SDP development.

4.3 Bug Causes Analysis

Figure 5 (a) shows this development ratios of bug causes after IT and Fig. 5 (b) shows its of SDP development. Major causes are detail design problems and coding problems as same as waterfall development but the top causes of this development are coding problems (the top causes of waterfall development are detail design problems). In this development, M/UT phase took only one week for each iteration and we evaluated programmers could not get rid of coding problems sufficiently in UT phase. It is also said that ratios of specification problems and basic design problems are larger than those of waterfall development. This was because we made codes in parallel with specification discussions and some specifications remained vague in M phase. In waterfall development, we reviewed documents of specification and basic design and could reduce these problems in design phases. To resolve specification misunderstanding problems, we provided a Cloud platform of middle iteration to operators and their development team, thus they could find critical problems in advance. Gathering all members in one office is difficult for middle or large development, it is important to provide developed software to operators and to reduce specification misunderstandings in early stages.

5. Conclusion

In this paper, we adopted and evaluated agile software development for a carrier Cloud service platform development. We could launch our production IaaS services in a half year after we started a resource management server development.
Therefore, we judge agile software development is effective for fast service launch. We compared and evaluated this development with previous waterfall development. It is especially said that automatic regression tests are important to repeat short term iterations. We could reduce 10-20% regression test efforts by automatic test cases using IT test data of previous iterations.

References


