An Automatic Testing Framework Used in Agile Software Development

Wang Fei Wu Xiao-yong Wang Yu-wei Li Jian-kun Sun wei
College of Information Technology
Beijing Normal University Zhuhai Campus
Zhuhai, China
wangfei171717@yahoo.com.cn, {wuxiaoyong, piaoerwang, lijiankun, angela--wei}@163.com

***Abstract*—automatic testing is very important for agile software development. There are some testing tasks can be finished by automatic tools, but most of them can not meet the actual needs of enterprises which is the direct reason of building an agile automatic testing framework named as Agilework. The structure, operation principle and building steps of the Agilework are descripted in details. The Agilework is a scalable framework which has high efficiency and flexibility. It can meet the requirements of the automatic testing in agile software development.**

Keywords-Agile Software Development; Automatic Testing; STAF; Keyword-Driven Testing

I. INTRODUCTION

The thought of agile development brings dawn to the software development as well as challenges to the software testing. While using the method of agile to develop softwares, it is necessary to run the whole set of smoke testing of the product at least once a day, which makes the testers feel panic about the heavy workload. Thus, automatic testing is necessarily needed [1]. The radical reason of using automatic testing is that it can save time and give the feedback of the testing result as soon and frequently as possible. So the testers have much more time to consider better designs of the testing use-cases and to take more explorative tests. [2]. There are many automatic testing tools, but most of them can not meet the individuation needs of enterprises, which is the direct cause of building an agile automatic testing framework named as Agilework. Theoretically, testing tools can be used to carry out some automatic testing, which can save costs of building new testing frameworks. However, disappointingly, the utilization of fixed testing tools， such as record and playback tools, in the practicing process is lack of reusability and flexibility [3]. Thus they could not meet the diverse needs of enterprises. So an integrated and pragmatic solution of automatic testing (ie. testing framework) is needed to achieve the goal of automatic testing in high qualities and high effitiency.

II. THE DESIGN AND IMPLEMENTION OF AGILEWORK FRAMEWORK

Testing framework is a collection of automatic testing standards, basic codes of testing scripts as well as a set of testing thoughts and convention. Testing framework is not only the basic codes of a group of standards and scripts of automatic testing, but also the combination of testing methods and practices. More and more software organizations and individuals prefer using their own logic to interpret automatic testing frameworks and establish suitable frameworks according to their own product properties [4]. The framework established in this paper mainly has following characters: First, it is based on the natural language when developing testing use-cases. Second, the Agilework supports distributed testing environment. Third, it supports the applications which are based on web. Forth, it supports multiple browsers. Those advantages are benefits from the following components which are integreted: Cucumber, STAF, and Selenium-WebDriver. Cucumber is an automatic testing tool whereby executing functional description text, which uses Gherkin as its descriptive language to create an action-driven development tool based on natural language [5]. Selenium-Webdrive is one of the open source project of OpenQA, whose purpose is to develop a cross-browser which is a contributed Web UI testing frame. The predecessor of Selenium-Webdriver which is also called Selenium 2 is Selenium, which developed into Selenium-Webdriver after combined with another open source project, Webdriver. Because it is free and easy to use, it attracts a lot of enterprises and developers. STAF (Software Testing Automation Framework) is an open-source framework which supports multiple platforms and language frames. STAF is about the reusable components and services, whose purpose is to make software testing much easier, especially make it easier to achieve the goal of testing automatically. In order to adapt for the process of agile testing, Agilework frame is aimed at developing testing and sorting testing report automatically, which means fewer staffs are needed. When testing cases are increased and changed, only a few changes are needed to adapt for the new situation, which sets free the staffs from the repetitive work. On the technical level, because of the needs of testing project, frame especially cares about cross-platform character, which is aimed at covering the control systems of Windows, Linux, Solaris and AIX and supporting the browsers of IE, Firefox and Chrome.

A. The structure of Agilework

As to the design of frame, in order to get better ability to work on different platforms, most testing modules run on JVM and use Juby as the main language. The reason of using JRuby is that it is a completely object-oriented interpreted programming language, which doesn't need to compile before running, moreover it has good readability, and achieves the goal to call java class seamlessly. The structure of Agilework frame is shown in Fig. 1:

Published by Atlantis Press, Paris, France.

Figure 1. The structure of Agilework framework

B. *The Runing Principle of Agilework Framework*

Fig. 2 shows testing environment includes Test Machine and Automation Server. Shown as process 1 in Fig. 2, Manual Trigger, Timer trigger, and Massage trigger are the three triggers to start a testing. Manual trigger allows users to run the testing when they need. Timer trigger allows users to set the time to run the testing. Message timer starts the testing through HTTP and Web Service. When finishing compiling a product of testing, continuous integration system will trigger the Scheduler Startup Testing shown as process 2. System will run cucumber to load and analyze fundational description file to call testing scripts. By calling testing modules (STAF and Selenium-Webdriver), scripts achieve the goal to do testing on remote testing machines shown as process 3, 4, and 5. After testing, testing report will be sent to the related staff by emails shown as process 6 in Fig. 2.

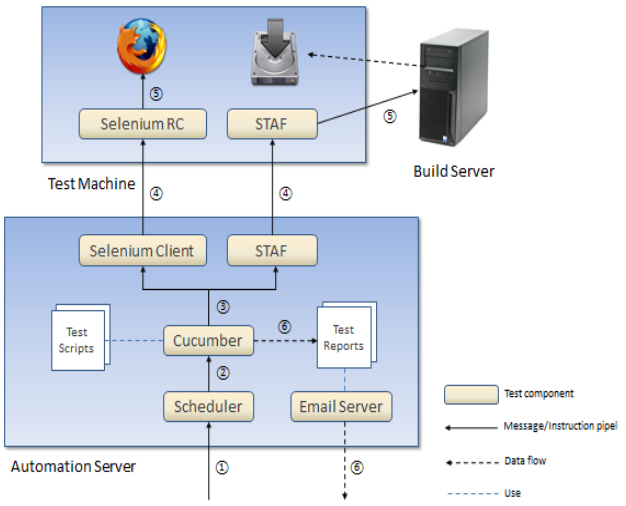

Figure 2. The runing principle of Agilework framework

C. *Building Agilework Framework*

First, integrated automatic testing environment, including an integrated script environment, deployment of Agilework framework and an automatic testing running environment should be established. RubyMine is selected as the debugging tool of script development in the Agilework. RubyMine is known as the most intelligent IDE of Ruby and Rails. As Fig. 3 shows, the code of the framework itself and the code based on the framework can be checked out after RubyMine has been downloaded and installed. And then the code of project can be imported into the RabyMine. The project includes many folders, such us comfig, core_infrastructure, features、resources, step-definitions, tmp, etc. Among these project folders, we focus on two of them: feature and config folders.because these two are of great importance. All of feature files for development are put into the feature folder and configuration files are put into the config folder. Various types of step are defined in the step-definitions folder, such as the step for executing commands, the step for clicking buttons and the step for writing data, etc. These steps are available for the use of feature files. The core_infrastracture includes many category files which give the definition of all components for the step-definitions using, such as the Button, checkbox and dialog components. Some resources files such as the mail templates and the licenses of the software are put into the folder of resource. The function of the tmp folder is similar to the resource folder's, however, some temp files which may be covered or adapted by other files almost every running time are put into the tmp folder.

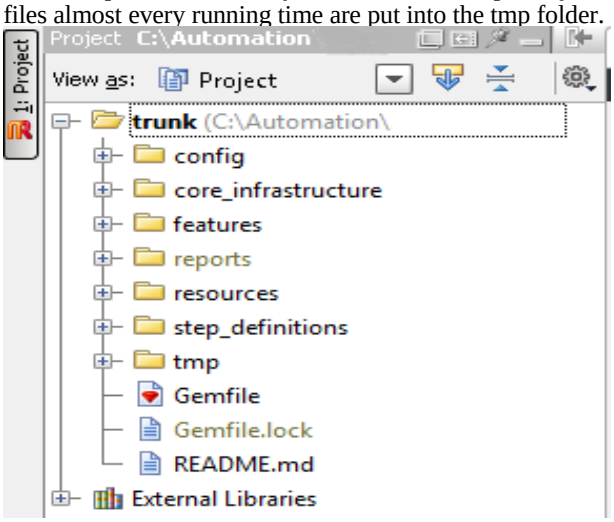

Figure 3. The folder sturcture of RubyMine in interface developing

Furthermore, a distributed-testing environment that is developed by virtual machine is needed. The STAF is used when communicating between VM and server as well as among the VM. In order to simplify the process of developing the environment on VM, all browsers are deployed on the server. Scripts use Selenuim-WebDriver to drive browsers to perform the relative operations, and use STAF to perform operations on other machines. After automatic test, a testing report is generated and sent to related staff through Email-Sender. STAF is used to

accomplish the communication among the machines, so a STAF is needed between the Automation server and VM. STAF is open-source and free, so it can be downloaded from its official website http://staf.sourceforge.net/directly. When a STAF is downloaded, it contains all the internal services, besides, all the other external services that will be needed also can be downloaded from this official website, then modify the configure file, staf.cng to add the relative external services into it. Before running automatic scripts, we should make sure that every machine's STAF is already started up and that they can communicate with others. We can use service ping to test whether it can communicate between two machines.

III. THE APPLICATION EXAMPLES OF AGILEWORK FRAMEWORK

A. *The core of Agilework framework -- the testing based on the testing scenarios*

By integrating the Cucumber into the Agilework, testers of automatic testing can not only use natural language to describe the user scenarios but also execute the scenarios to get the effect of the test. Cucumber can make the automatic testing codes more easy and direct to understand. Anyone who enjoys the relevant interest can use this function description to discuss questions. It can use the key words: if, when, and so to indicate attributions of these steps. 'if' often shows preconditions, 'when' express actions, the word 'so' shows verification conditions. How the Cucumber understand the descriptions inside the scenarios? Actually, script developers are needed to write the step definitions. Each step matches the relative step definition whereby regular expressions, then execute the content inside it. The codes inside step definition can perform relative operation according to descriptive steps of scenarios. The following is a testing use-case based on the scenarios: a scenario of a user installing the performance monitoring software A.

Function: install software A.

In order to use the lastly-buying performance monitoring system and manage my server, I, as the administrator of information service center, want to install this system well.

Scenario: install software A.

Assumed that I have put the installation CD into the CD-ROM, so when I click the installation icon, the installation interface should be showed. When choosing the "agreement" in installation interface, clicking the button 'next', inputting the route of installation and click the start-installation button, the finishing interface should be showed. When clicking the finish button, the installation interface should be closed.

Furthermore, the logs do not include "WARN, ERROR" errors.

It is based on text that Agilework accomplishes automatic testing and realizes the description of function. The advantage is that there is no need to use two sets of description of manual test and automatic test.

B. *Testing baseed on remote control system*

Agilework achieves remote control mainly through open-source frame STAF and provides an efficient and little resource occupied executive platform of reusable testing components, making testing components have the maximum of reusing. For example, the components of Files System, Process, and FTP can control the operation of machine's files, FTP and system processes. Besides some STAF own components, users can realize testing components of self definition according to interfaces provided by STAF. The following is an example of the description of how the framework can realize the distributed log check.

1) The design and realization of log-checking components

STAF gives the coding interfaces STAFServiceInterfaceLevel30 for self-definition components, which includes the methods that the developer must implement. The methods are as follows:

STAFResult init(InitInfo info)

STAFResult acceptRequest(RequestInfo info)

STAFResult term()

When the services are initiated, they will call init function to register the service, then clog itself and wait the request.when a request is coming, it will recall the acceptRequest function to create a new RequestInfo's example whereby request's commands and parameters, and introduce into the acceptRequest function as the parameters. Inside the acceptRequest function, users can realize the concrete functions. At last, term function will be called while uninstalling service, and developers should release resources at this point. According to Fig. 4, LogCheckService which is defined by users themselves are implemented from the STAF service interfaces. Concrete log analyzing logics are in the LogReader category, which can check it by setting log levels.

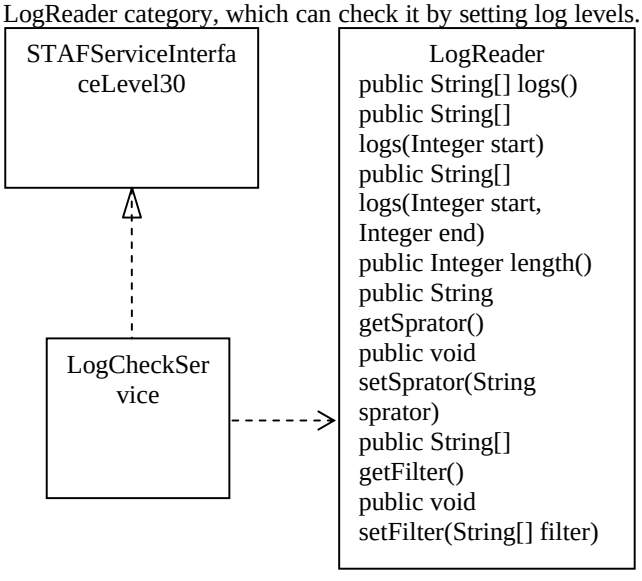

Figure 4. LogCheckService UML

2) The log-checking components and the integration of framework

All the services that are written by users will be deployed at each testing machines, and testing server will call the functions inside long-range service whereby STAF's command styles. Fig. 5 shows us that, by loading the file

STAF.jar, JRuby can control STAF process whereby codes to sent commands to remote testing machines. These machines check the log files whereby user self-definition services. At last, LogCheckService will return the result, and we can use the result to judge whether error appeared during the testing period.

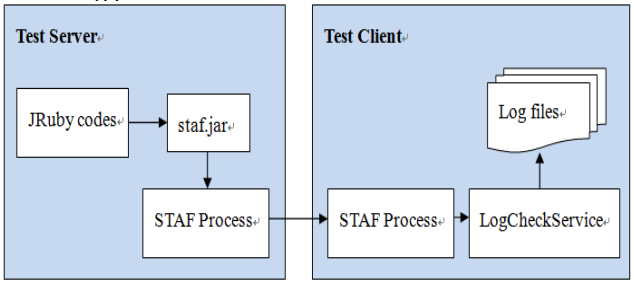

Figure 5. The invokeing process of STAF service

3) An Example of Log Checking Components in Agilework Framework

By using natural language to describe the users' usage scenario, it is possible to decrease the difficulties and cost of automatic testing. As shown in the Fig. 6, it uses STAF port to do remote invocation.

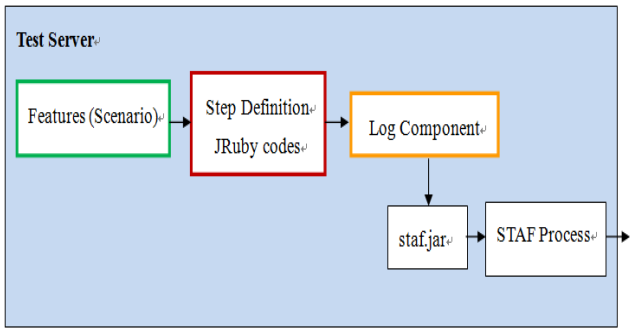

Figure 6. process of remote invocation

When the scenario run the step that the log does not contain the bug of "WARN ERROR", it will match the following step definition:

```
Then  do |severe|
      log_checker = LogChecker.new()
      log_checker.check(severe)
end
```

At last, it will create an instance for the log testing category and call the check method to test according to the level of the serious situation. Testers only need to correct scene description file to adjust the serious level of the log according to the needs without revamping the the test codes.

The statements shown above use the prosses of testing log file to explain the operation process of the Agilework automatic testing frame. Becasue of the limition of the pages other cases are not provided.

IV. CONCLUTION

In conclusion, because it integrates the modules with Cucumber, STAF and Selenium-WebDriver, testing frame Agilework has the characters of testing cases based on natural language, supporting distributed testing environment, supporting the testing based on web applications, and supporting multi-browsers. Automotic testing which are successfully and effectively applied in agile software can significantly improve the agility of testing.

REFERENCES

[1] Shen Lei, Shen Bei-jun. Research and Practice of Agile Methodology [J], Computer Engineering 2005(7)

[2] Zhou Fei-yu. Design and Implementation of an Automated Testing Platform [D], Beging jiaotong university 2009

[3] NI Ming, HUANG Ping. Script-based Component Test Automation Framework [J], Computer Engineering 2010(6)

[4] Hu Hui-fen, The Research and Application of Software Test Automation Framework [D], Xidian University 2008

[5] Cucumber. http://cukes.info/