Development of a Continuous Integration Framework

Utku Acar
Vestel
Manisa, Turkey
Utku.Acar@vestel.com.tr

Burak Öyke
Vestel
Manisa, Turkey
Burak.Oyke@vestel.com.tr

Ömer Özdemir
Vestel
Manisa, Turkey
Omer.Ozdemir2@vestel.com.tr

***Abstract*—This project focuses on preventing time loss which is caused by possible mistakes or incompatibility between MediaTek and Vestel companies. Tools which are Subversion (SVN), Git, Bash Script, SMTP, and JUnit5 have been used to apply the integration framework. Because of the integration framework, the system detects this kind of issue earlier via building projects each night and lets the co-worker know which files have caused it via processing build logs each night and implementing our integration framework into the system.**
We have eliminated the lack of automation in our server by creating the Integration Framework and adding new features to automate daily tasks as much as possible.

***Index Terms*—Software Testing and Analysis, Integration Test, Test Automation, Unit Test, Bash Script Test, Automated mailing.**

I. Introduction

The software is used in more and more systems every day. The widespread use of software increases the complexity of the software. Due to the complexity of the systems and the increase in the complexity of the software, there are large software teams within the companies and the software works are carried out between the companies as shown in Fig. 1.

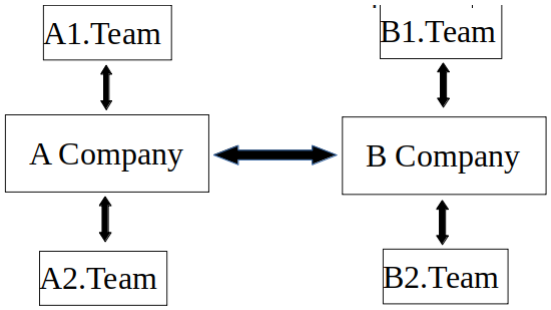

Fig. 1. Software Data Flows

Conflicts may occur in the software committing process between companies or teams. Because of this, Daily build and integration frameworks have emerged and investigated co-worker performance in the Android TV application domain. That's why prevent time loss which is caused by possible mistakes or incompatibility between MediaTek and Vestel companies.
It manages the changes made to the files over time and these changes are stored in a database with version control systems. It allows us to revert to the old code record in cases where we encounter an error in the project. While the Vestel Android Software team uses the Git version control system with the other company, they use SVN among team members. They aimed to save time to detect the source of the error in case of a possible error. A user-like agent has been created on our corporate team server that acts as an integration framework With some changes described in the next section.
An integration framework that can be built from scratch was introduced for all projects, and if there were any errors while generating the source code, it was identified which commit caused it and notified by email to the committer. In addition, the ability to test system components as a unit has been introduced. In the next section, brief background information on the tools and techniques used in our project is presented.

II. Background

A. Tools

1) SSH Server: SSH is a network protocol that provides a secure connection between a computer and a remote computer. The protocol has encryption capabilities, this feature enables us to securely connect to our machine even in an insecure network [1].

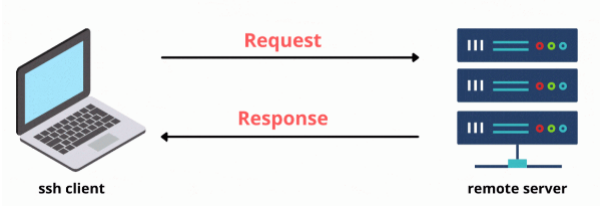

Fig. 2. SSH Connection [2]

Vestel's Android Software Team maintains all projects on an SSH remote server and performs their work there.

2) SVN: SVN is an open-source, free, and centralized version control system that could be by everyone. It is built to execute simple to complex tasks efficiently and successfully. It was built to coordinate the programmers' work. Version control enables team members to work simultaneously and be managed in the same workspace [3].

3) Git: Git is the most common advanced version control system in use today. Linus Torvalds, the legendary designer of the Linux operating system kernel, officially developed Git in 2005. It is a well-established, continuously maintained open-source project. Git is used for version control in an incredible number of software projects, both open-source and commercial [4].

4) Software Test Methods: Testing is the process of manually or automatically evaluating a system to verify that it meets specified requirements or to identify differences between expected and observed results. Software testing includes dynamic verification activities to meet the expected behavior of software from an infinite number of work areas, with a limited number of appropriately selected tests [5].

These arrangements have been made by different models. You can see an example testing model "V-Model" in Figure 3 below.

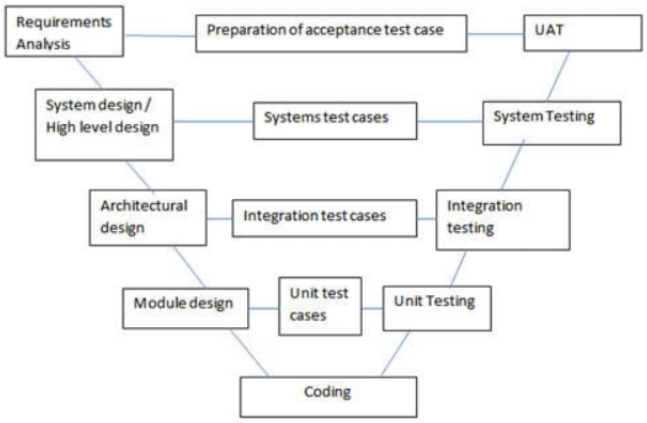

Fig. 3. Software Test V-Model [6]

We have used 2 Testing methods which are Unit and Integration Testing. We used unit testing directly by JUnit and integration testing non-directly since we are building the code with the changes made on that day.

a) Unit Testing:

Unit testing is a testing method that provides us to test the code on the atomic scale. It also forms the basis of this process in terms of early detection and correction of errors. In this micro-scale test, special functions or code modules (functions, data structures, objects, etc.) are tested. This testing is done by programmers, not testers, and it is necessary to know the details of the program code and its internal design. It is quite a difficult test if the application code is not in a very well-designed architecture. [7]

b) Integration Testing:

It is a test technique to test whether different components of an application work together in harmony. They can be in the form of components, modules, standalone applications, and client/server applications. This type of testing is especially applied in testing client/server applications and distributed systems [8].

5) Shell Scripting: A shell script is a computer program meant to be run by the Unix shell, a command-line interpreter. Scripting languages include different varieties of shell scripts. File manipulation, program execution, and text printing are common tasks carried out by shell scripts. The term "wrapper" refers to a script that creates the environment, launches the application, and does any required cleaning or logging [9].

6) Log Analysis and Manipulation: Log analysis is the process of converting raw log data into information that may be used to solve problems. Log analysis is used to detect inefficient parts, simplify workflows, automate activities, emerge powerful analytical languages, and discover lingering issues [10].

III. The Project Work

A. Working Scheme of Integration Framework

An integration framework that can be built from scratch was introduced for all projects, and if there were any errors while generating the source code, it was identified which commit caused it and notified by email to the committer. In addition, the ability to test system components as a unit has been introduced.

The project consists of three phases. The stages of the project are listed below.

- Build Part
- Data Process and Mailing Part
- Testing Part

These stages can be seen in the flowchart of the Integration Framework in Figure 4.

1) Building Part: MTK and Vestel have different repositories to save codes. If MTK shares software with Vestel, Vestel merges with its own code blocks. This is where the integration framework gathers both MTK's files and Vestel's files every day in one server account which is called the "atvnightlybuild" account. All files merge with each other on the server. Afterward, Full build commands execute with each project. This operation takes about 10 hours for overall projects. Results of the full build which are called logs are gathered in a document. The decision mechanism decides according to full build logs and passes to the data process mailing part.

2) Data Process and Mailing Part: After controlling build logs for an error and if an error is encountered during compilation, the location of the error and who made it are determined. The location of the error is inferred according to the full build log document. Committers who make a mistake are found according to the similarity between the location of the erroneous file and the committed file location in every commit and then send a mail about the related errors to the related committers with possible erroneous file locations. Also, there is a mechanism that mails build results to everyone in

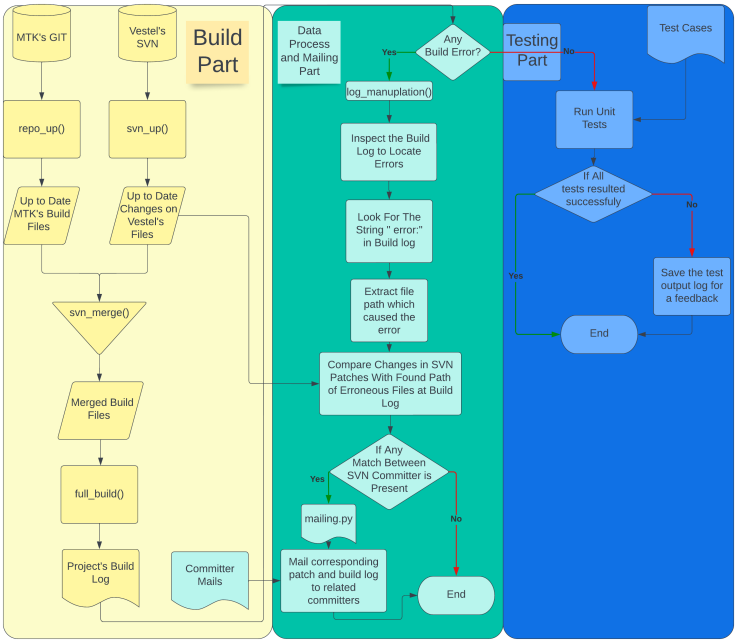

Fig. 4. Flow Chart of Integration Framework

our team which includes results for all projects in our system as soon as the building part has been completed.

3) Testing Part: When part A and part B have been completed and the result of the full build is successful, some expressions and returned values like if the Backlight value has been taken correctly from the Config.ini file or not by the getBacklightValue method are checked by the unit test operation which is made by the JUnit module. The results of these tests are included in daily build mail as an attachment. These test cases are also pushed to our SVN server.

B. Detection of Origin of Issues

The Integration Framework also tries to build just MTK's Repository files if the merged version of the MTK's Repository files onto Vestel's SVN files fails. This process is used to determine the origin of the related issue. If the build with just MTK repository files also fails this means that the problem originated the MTK's side.

C. Detection of Outdated image and APK files

Our System also has a feature that checks for outdated images and APKs by comparing these files with the resultant files of the build. If there is any difference in these files, the system logs a warning as "NOT UP" for these outdated files into the total results log and if there isn't any difference between files, the system adds a log to the total results log as "UP".

IV. Results & Discussions

A. Detected Commit Issues

Our system has detected commit issues since we implemented it on the server. After our system indicated and reported these commit issues to related committers, committers took action swiftly and solved these commit issues. The solution of these commits saved us nearly 10 days to get Google Signed Build for Android & Google TV.

B. Detected MTK's API problems

Our system has detected some API issues on MTK's side. After some days after we implemented our project to the server, on some projects, we have seen that there was a build failure with merged software then the system also tried to build just MTK's files, and the build also failed and logged into a summary. This means that this issue has been caused by MTK and not by our team. We have also checked the project's build logs and found that this issue has been caused by MTK's API. We mailed this issue to MTK and saved about 10 working hours for 3 people in our team to address and debug that issue.

C. Detected Outdated APK and Images

Our system has detected there were some outdated APKs and Image files. In some of our projects, we are working with Prebuilt APKs which means that MTK is using those APKs to merge our fixes and changes with theirs. If we do not commit the newest APK files which are created with our latest source code to our SVN, the APKs at our SVN become outdated and which means our fixes are not included APKs and this propagates through weekly commits and results in reopening fixed bugs in our bug control system even if the latest source code has our changes. Because Google Signed Build has been made by our APKs instead of the source code. Also, this signing process takes about 10-14 days so this detection helps us to save that much time and working power.

D. Project Build Prioritized Scheduling

Our shift starts at 8 AM at our company and since our building process takes about 1.5-3 hours for each project and we have 7 projects and some of them have more than 1 branch, we need to have at least 10.5 hours. So, starting the nightly build at 12 AM every day would not be enough to build all projects until starting time of our shift so we decided to prioritize projects due to their importance in market size and the currently open number of bugs. We have done this by scheduling projects. While more important projects build every day, projects with less importance build once in 10 days or etc. Also, we have set the system to be started at 8 PM instead of 12 AM.

V. Related Work

Our project both differs and intersects with other projects in the same field in using techniques, technologies, and tools.

LHCb Nightly Build System: This system had been used for the Large Hadron Collider Beauty (LHCb) project at CERN Laboratories and it is based on an open-source integration tool named ”Jenkins” unlike our system. They build their projects on Linux OS like ours. Also, they introduced a concept of ”slots” which they use for starting builds with different configurations on different platforms. We have used a similar approach by using customizable default parameters in each method in our driver script. They start their tests just before starting a build of the next project. We are following a similar approach with starting our Junit tests just after the build of each project finishes. They have a configurable log parser like ours. They also have a pool of build machines so they are capable of distributing a building load between these machines and building more than one project in a parallel manner while we are able to build our projects on just one server and in a serial manner. Each step is named a Jenkins job and each of them corresponds to a method in our shell script. They check out their codes from GIT and store them sometime after the build was completed while we are just keeping pkg and apk files. Their scripts are based on Python and their code base is also written in Python while our scripts are written in bash mostly and also Python for the parsing and mailing part and Java for the unit testing part. They are checking out their latest scripts from GIT each day just before the builds so are but we are using SVN. They are uploading their build logs and results to their databases while we have chosen mailing operations for sending build logs and results. [11].

ATLAS Nightly Build System: This system is a derivative of the software part ATLAS project at CERN Laboratories. Their goal was to implement new patches which contain the latest changes in the code-base to the previously working software with testing by unit and integration tests which is exactly the same as our goal. Their code base consists of nearly 5.5 million lines of code while 75 percent of these lines have been written in C++ and the remaining 25 percent have been written in Python by 1000 developers. They are using a powerful control tool named ”NICOS (Nightly Control System)” and this tool provides great computation power to run unit and integration tests in parallel while we are running our test cases without using any tool because of restrictions of our server environment. This tool also detects problems and coding defects and shows them on the tool’s website. They are using auto-mailing to the related developers for building results and problems just like we do while they are using NICOS and we are using our Python script for these tasks. They have also designed their code in a modular way to create simple and user-friendly methods such that each of the methods works stand-alone too just like we do in method structure in our driver script. Their initial complete build time was 22 hours in 2005 which was not enough to call it a ”Nightly Build” so they increased their computational power by adding 45 more nodes and improving hardware specs from 1 GB of RAM to 16-24 GB RAM per machine while the nightly branch number also increased over 7 years from 1-2 to 45-55. Our initial complete build time was about 7 hours for 4 projects and this was enough to call it ”Nightly” but later we implemented 5 other project branches as well and the total building time increased to 13 hours and this time was unacceptable. So we reduced this time to 10 hours by using prioritized scheduling and starting the build at 8 PM instead of 12 AM because we are limited by hardware. They are thinking to implement starting ”On Demand” nightly jobs instead of every day. Since our driver script provides such an ”On Demand” nightly build initiation at any time, we are capable to do it currently [12].

Geant 4 Nightly Builds System: Geant4 is a set of tools that are used to mimic the behavior of particles while they are getting through a matter which is a project at CERN. Since the project is multi-national and developers are from different companies, patches to the code base should be tested frequently. The old version of the Geant 4 testing system consists of shell and Perl scripts and only works on Unix platforms and Mozilla TinderBox used for getting test results. This system has been migrated to the Large Hadron Collider (LHC) application Night Build System rather than creating a new night build system. This migration helped them about building their projects on Windows and Mac OS in addition to UNIX platforms. They used CVS while we are using SVN for the version control system. Since they have different branches for different purposes, commits are separated with a tag collection system to test specific changes rather than testing the latest code while we are running all test cases without any cherry-picking because our team is living in the same time zone and no need to worry about developers living at different time zones. They have also a connection between their repository and MySQL server to process tags and decide which project should be built or tested while this decision depends on prioritized scheduling which can be modified manually for now at our side. They have also error detection and feedback mechanism like ours. The data processing part

of this system is similar to ours. After the process, they are keeping results on their server for 7 days, while we are keeping these results for 1 month. Since the platform they merged on already has a link to the database, they have decided to use this database to keep their data organized and accessed easily while we are thinking of using a database if required sources provided to make the system more robust [13].

VI. Conclusion

The lack of automation in our servers has been eliminated by completing this project. The Integration System has been made and developed each day due to our company's needs and requests from coworkers. Since we made this system in a modular way every function can be used standalone, which provides us the ease of use for everyday work in our team. The test case number will also increase as time passes. As we continue to add new features to the system to automate the daily work as much as possible, this tool will become even more powerful.

References

[1] UCL, "What is SSH and how do I use it?," Information Services Division, 14-Oct-2020. [Online]. Available: https://www.ucl.ac.uk/isd/what-ssh-and-how-do-i-use-it. [Accessed: 19-Dec-2022].

[2] Karthick, "What is SSH and how does it work?," Geekflare, 20-Nov-2022. [Online]. Available: https://geekflare.com/understanding-ssh/. [Accessed: 19-Dec-2022].

[3] "SVN tutorial," www.javatpoint.com. [Online]. Available: https://www.javatpoint.com/svn. [Accessed: 19-Dec-2022].

[4] Atlassian, "What is Git: Atlassian Git Tutorial," Atlassian. [Online]. Available: https://www.atlassian.com/git/tutorials/what-is-git. [Accessed: 19-Dec-2022].

[5] "What is software testing and how does it work?," IBM. [Online]. Available: https://www.ibm.com/topics/software-testing. [Accessed: 19-Dec-2022].

[6] Z. Iqbal, "What is V-model and advantages and disadvantages of this model," Software Testing, 24-Jul-2015. [Online]. Available: https://testingsorttricks.wordpress.com/2015/07/24/what-is-v-model-and-advantages-and-disadvantages-of-this-model/. [Accessed: 19-Dec-2022].

[7] O. Moradov, "Unit Testing: Definition, Examples, and Critical Best Practices," Bright Security, 24-May-2022. [Online]. Available: https://brightsec.com/blog/unit-testing/. [Accessed: 19-Dec-2022].

[8] TESTRIG, "Top Different Types of Software Testing in 2022 — Testrig Technologies," Top QA & Software Testing Company, 11-Apr-2019. [Online]. Available: https://www.testrigtechnologies.com/different-testing-types-used-by-software-quality-assurance-companies/. [Accessed: 19-Dec-2022].

[9] "Shellscript," Wikipedia, 16-Jan-2008. [Online]. Available: https://en.wikipedia.org/wiki/Shellscript. [Accessed: 19-Dec-2022].

[10] S. Alspaugh, B. Chen, J. Lin, M. A. Hearst, R. Katz, and U. C. Berkeley, "Analyzing log analysis: An empirical study of user log mining," Usenix.org. [Online]. Available: https://www.usenix.org/system/files/conference/lisa14/lisa14-paper-alspaugh.pdf. [Accessed: 19-Dec-2022].

[11] M. Clemencic and B. Couturier, "A new nightly build system for lhcb," Journal of Physics: Conference Series, vol. 513, no. 5, p. 052007, 2014.

[12] A. Undrus, "Evolution of the atlas nightly build system," Journal of Physics: Conference Series, vol. 396, no. 5, p. 052070, 2012.

[13] V. Diez, G. Folger, and S. Roiser, "Geant 4 nightly builds system," Journal of Physics: Conference Series, vol. 219, no. 4, p. 042038, 2010.