The Utilization of Automated Tools in the Automated Continuous Integration Practice Case Study: Medical Record Application

Eka Trisno Samosir[1], Hernawati Samosir[1], Inggriani Liem[2], Yudistira Dwi[2]
[1]*Del Institute of Technology, Sitoluama, Laguboti, Toba Samosir, North Sumatera – Indonesia. 22381.*
[2]*Bandung Institute of Technology, Jl. Ganesha 10, Bandung, West Java – Indonesia. 40132.*
eka.samosir@del.ac.id

***Abstract*—Continuous integration (CI) is a software development practice that is carried out in a team by dividing modules among the team members and integrate team's work regularly. Developers who are using CI practice manually will require more effort for the whole work integration compared to those who work in a team and integrate their work regularly. The application of toolset in CI practice will enable the developer to work easily. The CI practice that uses toolset is known as automated CI. The automated CI practice consists of version management using version control system tools, code program testing using testing tools, build execution using build tools and module integration practices using CI tools. From all of the automated CI practices, the focus of this research is the execution of build and the integration of modules manually and using automated tools. The significant differences from using automated tools in automated CI practice are the decrease of unnecessary effort spent by developers on the execution of the build using the build tools and the elimination of the integrator role by using CI tools that finally results in a more efficient performance of the developers.**

***Index Terms*—Build Tools; Build; Automated CI Tools; Continues Integration.**

I. INTRODUCTION

Software engineering is a strategy in developing software which includes processes, methods, and tools [1]. The life cycle of software development consists of requirements, design, code, testing [12,13], deployment, and maintenance [2]. In the large scale of software development, a new feature will be added incrementally. This process will travel through the life cycle of software development for several times until the product with a certain version is released. According to the survey of software development in some companies, maintenance costs are greater than other processes [3].

During software development process, software specification tends to change according to the users' requirements and they urgently need it. Basically, a software is built with attention to two main focuses [2]; cost and development time in which those are related to each other. The duration of software development itself is effected by various aspects including the failure in the integration process. The bigger the problem is the longer it takes to fix it, especially if it is done manually.

Automation is the key to do the same and repeating process. By implementing automation, the time to build, deploy and test the software can be minimized and shortens. In term of integration, the automation process can be used to integrate the software regularly which can also reduce the risk of software failure and improve the quality of the software [4]. This integration process is done using the implementation of Continuous Integration (CI).

CI is a software development practical that integrates the developer's works regularly to find the problem in the integration process immediately [5]. CI implementation using toolset is known as automated CI. The automated CI will ensure the software to run in every modification [9].

If the problem is encountered during the integration process, the appointed team can fix it immediately. For instance, the team that uses automated CI can effectively detect bugs faster, produce software with fewer bugs, and reduce the cost and time to amend the software compare to other team with no implementation of automated CI [4].

The implementation of automated CI comprises version control system tools, testing tools, and automated tools. In this research, the author focuses more on the implementation of automated CI, particularly about the automated tools. The use of automated tools will be implemented in a medical record application called Medrecapp built using Java programming language.

II. ANALYSIS AND METHOD

A. General CI Concept

CI is a software development method where the task is divided into modules and done by a team. Each module will be distributed to each member of the team and integrated after finished [8]. In manual CI, there is no toolset is used to support the integration process which will make the development process vulnerable to a particular problem. Some activities that are done in manual CI including version controlling, code testing, build execution, and module integration.

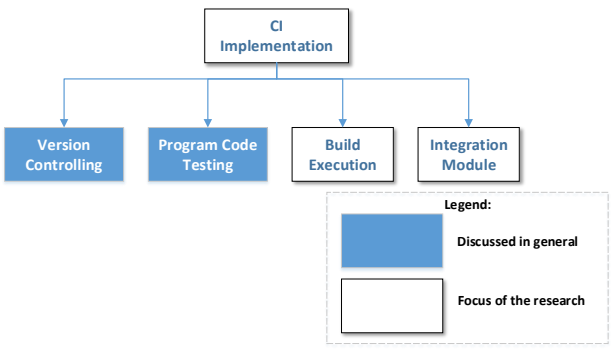

Figure 1: General CI Implementation

a. General CI Concept of Manual Build Execution

In the general concept of manual build execution, integration is required to trigger the application archiving and deployment process. The application archiving process consists of files that are ready to be used. Generally, the trigger of build execution is done by an integrator in the integration machine.

Before triggering the process of creating the application, an integrator needs to integrate and test all the correct module versions from developers. The build process includes the trigger execution of all testing drivers, the trigger execution of all GUIs, the trigger execution of generating application package, the trigger execution of application package, and trigger deployment of the application package to the customer environment. Those trigger processes are done solely by a integrator every time a member integrate it manually and repetitively.

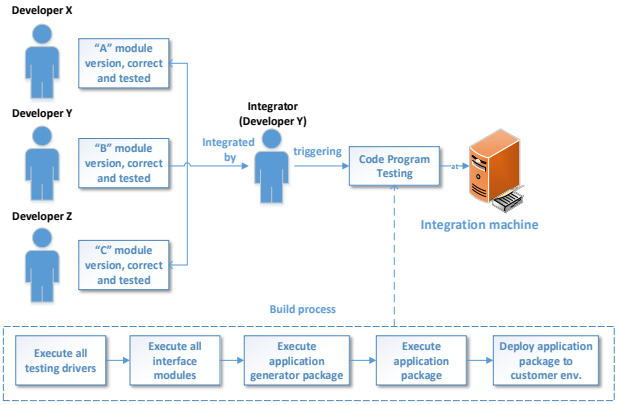

Figure 2: Manual build execution

b. General CI Concept of Integration Module

As mentioned in the previous paragraph, the integration process requires an integrator to integrate all the modules from developers and ensure the application has zero problems. The integration is started by triggering the testing drivers, GUI modules, generate application package, and application package execution. The aim of these trigger processes is to verify whether the functional requirement of the software has fulfilled. If the integrator encountered any problem, all developers should be notified immediately.

Successfully integrated modules will be generated into an application package and ready to be deployed to the customer environment. Afterward, an archive process will be done to acquire the histories from the application packages. The archive will be used as the milestone of the software development progress.

B. General CI Concept Using Toolset

The idea of using toolset in CI is to automate the integration process as mentioned in the previous description instead of using an integrator to make the process more efficient. In this toolset, the focus will be the build execution using build tools and module integration using CI tools.

a. General Concept of Build Execution and Build Tools

By using build tools, the code testing [11] and archiving process stored in local can be automated. In order to do the automation process, developer team needs a build script. A build script consists of some targets and tasks that will be executed by the build tools. Basically, a build script is created to synchronize the workflow of each team member in local machine and automate the build process that will be done by an integrator in the integration machine. The addition of target testing on every level of automated build covers private build, integration build, release build into build script is not a mandatory but it can minimize the problem.

The build script is executed by build tools on the local machine of each team member called private build. To synchronize all the workflows of the team member, a team needs to define the target and task of build tools. A target may consist of several tasks and may depend on another target. Generally, several targets in the private build includes code execution testing and module versioning storage which has modified into the local repository.

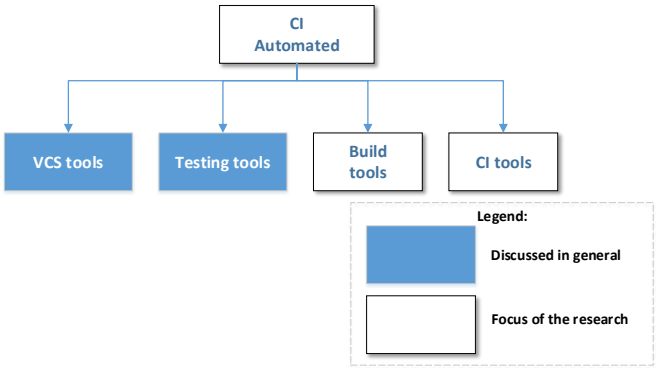

Figure 3: General implementation of automated CI

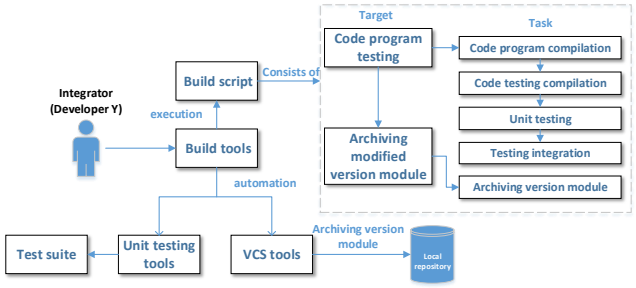

Figure 4: Private build execution

To automate all activities that will be done by the integrator in the integration machine, a team needs to define the target and task in the build script that will be executed by build tools. Build script which is executed by build tools in integration machine to generate the application package called integration build. Basically, target in integration build includes code testing execution and generate application package.

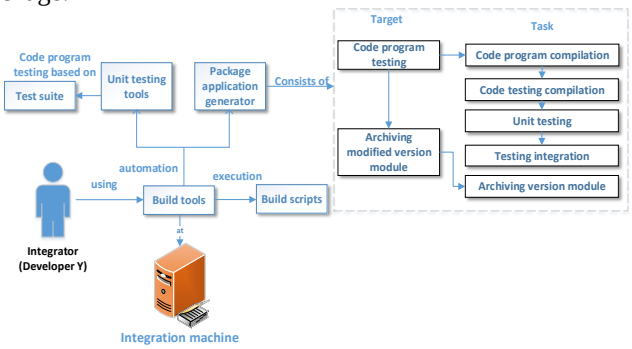

Figure 5: Integration build execution

Application package from integration build can be tested and deployed to customer environment automatically. To

automate that process, a team needs to define the target and task in the build script that will be executed by build tools. Build script that is executed by build tools in integration machine to deploy the application package called release build. Target in release build includes application package testing and application package deployment to customer environment.

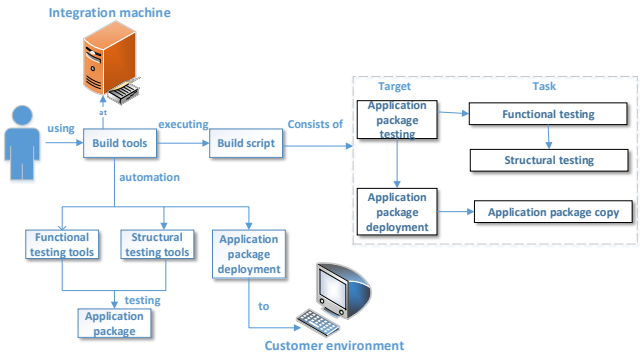

Figure 6: Release build execution

b. General Concept of Integration Module Using CI Tools

The integration process using CI tools in integration machine will not require an integrator to use the build tools because the build tools can be automated and scheduled. CI tools also can retrieve the latest module version from main repository automatically based on the schedule. That testing will be done on the integration machine based on the testing code saved in each team member in the main repository. By using CI tools, an integrator in the integration machine will not necessary anymore. If any problem occurred, the system will notify all the team members directly.

The archiving process will also be done automatically in the integration machine when the execution is succeed. CI tools will also notify the progress of the software development to each team member automatically.

III. IMPLEMENTATION

Medrecapp is a desktop application which is built using Java programming language. It consists of nine modules such as Specialize, Insurance, Patient, Staff, Nurse, Doctor, Action, Medical Record, and Service Action. Each module of Medrecapp includes DAO class (Data Access Object), entity classes, GUI classes, interface classes, services classes, and model table classes. The modules dependency is explained by this picture.

Manual implementation of Medrecapp without using CI tools has explained in the method sections. So in this section, the focus of explanation will be the implementation of automatic CI.

A. Implementation of Medrecapp Using Toolset

The development of Medrecapp application with CI using toolset includes control versioning using VCS tools [10], code testing using testing tools, build execution using build tools, and integration module using automated CI tools. However, the focus will on the last two parts; build execution using build tools (Ant) and integration module using CI tools (Jenkins) [7]. The rest of the toolsets have already done by Yuanita [5] and Fachrul [6].

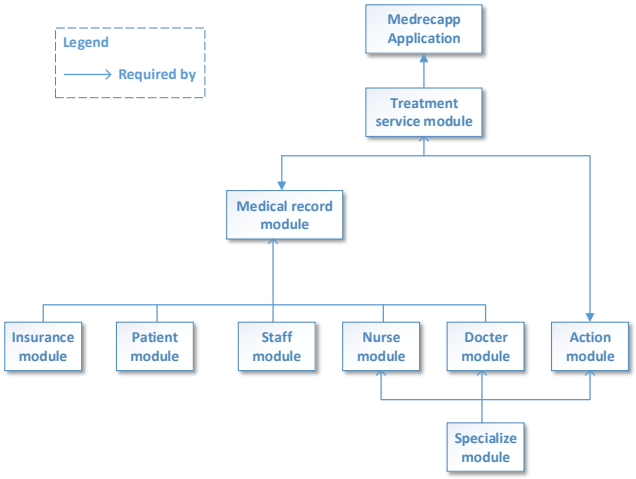

Figure 7: Dependency diagram among modules in Medrecapp

a. Implementation of Build Execution Using Build Tools

Build process in CI uses Ant as the build tools. The process of code testing, as well as module versioning storage in the local repository, can be automated using a build script that is built by the developer team to ensure the workflows of all team member synchronized in the local machine. A build script consists of some targets and tasks that will be executed by Ant. The team should automate the build process that will be implemented by the integrator in the integration machine.

The executed build script on the local machine of team member by Ant is called private build. As stated before, targets and tasks should be defined by the team where each target consists of several tasks and depends on another target. An example of the target in the private build is the code testing execution.

As seen in Figure 8, Hernawati uses build tools (Ant) to execute build scripts with code testing target which consists of 11 tasks.

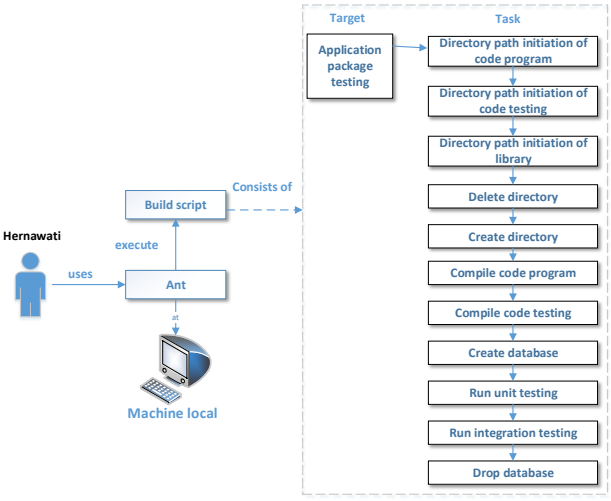

Figure 8: Private build execution

b. Implementation of Integration module Using CI Tools

According to Figure 9, CI tools (Jenkins) in the integration module will replace an integrator in the manual system to execute the build script. Jenkins will use Ant to execute the build script on the integration machine. A team will only require scheduling the build execution and then Jenkins will execute the build script accordingly.

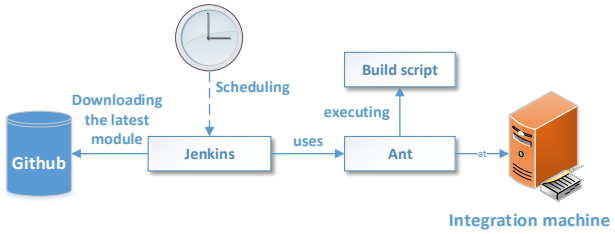

Figure 9: Build script execution scheduling in the integration machine

In every integration build execution, integration machine will test the code and application package automatically and notify the occurred problem to each team member which are all done by Jenkins. Jenkins execute the build script by using Ant.

CI tools can also automate the archiving process of the application package. Jenkins uses Ant to execute build script and store the archives in the integration machine as shown in Figure 11.

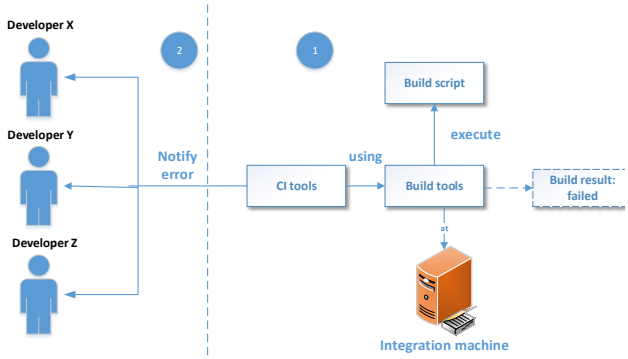

Figure 10: Automated notification system

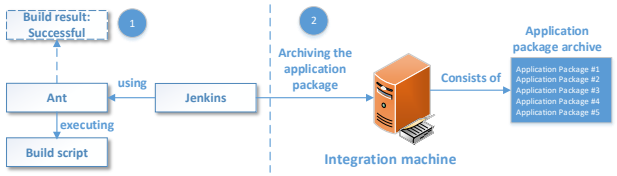

Figure 11: Automated archiving process

Following the previous process, Jenkins will create the progress report of software development in the integration machine and send it to the team members.

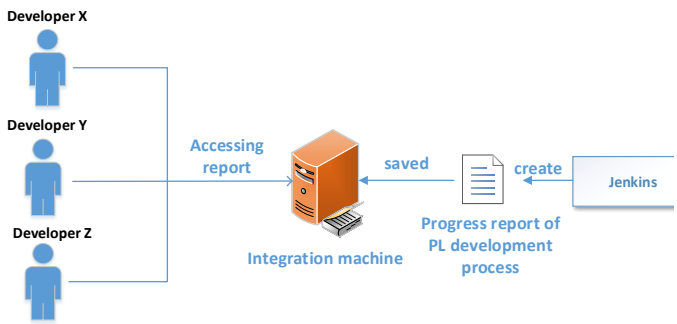

Figure 12: Automated progress report of software development

IV. Conclusions

The conclusion from the implementation of build execution and module integration manually as well as automatically using automated tools in Medrecapp application is:

1. The difference between manual build execution and build tools in Medrecapp application is on manual build execution, the developer needs to run several triggers, such as trigger in the testing driver, GUI in all modules, and generate the application. Whereas in build tools, the implementation of build execution will use Ant. Ant will execute the all the triggers automatically in integration machine.
2. In term of the integration process, the manual build execution requires an integrator to notify developers if a problem occurred. While using build tools, the integrator will be replaced by CI tools, called Jenkins. Jenkins will automate the build execution in integration machine as scheduled, notify the problems to developers, build the archive for the Medrecapp application in integration machine, and create a progress report of the Medrecapp application's development.
3. The framework to implement automated CI consists of two parts:
   i. Preparation
      - Divide the development modules into several parts
      - Create commitment before building the software where developers must store the codes accordingly and fix the occurred problem immediately after testing
      - Prepare the integration machine
      - Install toolset to support the automated CI which includes VCS tools, automated testing tools, automated build tools, and automated CI tools.

   ii. Process
      - Create the build script
      - Define dependency between targets and build script
      - Define the main target on build script
      - Set the schedule to invoke the build script by using CI tools

Acknowledgements

The author would like to thank you the Del Institute of Technology (Institut Teknologi Del) of North Sumatera for the full support in publishing this research.

References

[1] Glenford J. Myers, 2004, The Art of Software Testing Second Edition, Hoboken, John Wiley & Sons.
[2] Pilone Dan dan Russ Miles, 2007, *Head First Software Development*, USA.
[3] Kaner, Falk, Nguyen, 1999, Testing Computer Software, Second Edition (Wiley Series), USA.
[4] Humble, Jez dan David Farley, 2010, Continuous Delivery: Reliable Software Releases through Build, Test and Deployment Automation (Addison-Wesley Signature Series), USA.
[5] Hijriyah, Yuanita Annisatul, 2014, Penggunaan VCS tools dalam praktik automated Continuous Integration pada studi kasus aplikasi rekam medis, Program Alih Jenjang D3 ke D4 Teknologi Informasi

Kesehatan, Institut Teknologi Bandung, Bandung.
[6] Muhamad, Fachrul Pralienka Bani, 2014, Penggunaan testing tools dalam praktik automated Continuous Integration pada studi kasus aplikasi rekam medis, Program Alih Jenjang D3 ke D4 Teknologi Informasi Kesehatan, Institut Teknologi Bandung, Bandung.
[7] Meet Jenkins. (Online). URL: https://wiki.jenkins-ci.org/display/JENKINS/Meet+Jenkins. Accessed in 10 October 2013.
[8] Fowler Martin, 2006, Continuous Integration. (Online). URL: http://martinfowler.com/articles/continuousIntegration.html. Accessed in 10 October 2013.
[9] Duvall, Paul M., Steve Matyas and Andrew Glover, 2007, Continuous Integration: Improving software quality and reducing risk (Addison-Wesley Signature Series), USA.
[10] Somasundaram. Ravishankar, 2013, Git: Version Control for Everyone, Birmingham B3 2PB, Packt Publishing Ltd.
[11] Glenford J. Myers, Tom Badgett, Corey Sandler, 2012, The Art of Software Testing 3rd Edition, Hoboken, JohnWiley & Sons.
[12] Pressman. Roger S, 2001, Software Engineering: A Practitioner's Approach Fifth Edition, New York, The McGraw Hill.
[13] Peter A. Vogel, An Integrated General Purpose Automated Test Environment, ftp://192.67.63.1/pub/cite/vogel-cite.pdf. Accessed 3 May 2014.