Open Affect Project

Eclipse Plugin

Elisa Nannini, Misha Cattaneo, Loïc Bassang
Software Engineering, Haute Ecole Spécialisée de Suisse Occidentale
Lausanne, Switzerland
e-mail: elisa.nannini@master.hes-so.ch, misha.cattaneo@master.hes-so.ch, loic.bassang@master.hes-so.ch

***Abstract*—This document contains a résumé of the Open Affect Project that aims to measure the emotional state of a developer during code writing. Project team also used a continuous delivery pipeline to produce the builds of the Eclipse plugin.**

I. Introduction

In this project, our team developed an Eclipse IDE plugin that acts as client for an Open Affect server. The main goal of our project is to question developer's emotional state randomly in a day time frame. In return, we have a statistic on the best moments of the day for a developer to write some code, based on his emotions. We think that this project could show when the developer's job can be improved and how bugs in software or unexpected problems can influence his emotional state. We used the SCRUM method to help ourselves in the continuous delivery process. The continuous delivery pipeline was also built in Jenkins, the software we decided to use to provide automatic compilation.

II. How does it works?

The developer is questioned using a popup dialog with three radio buttons. These radio buttons permit to make one of 3 choices: "Sad", "Indifferent", "Happy". The developer selects a radio button and then clicks on a push button to confirm the choice. Afterwards the popup dialog disappears. The emotional state of the developer is sent to the Open Affect server that stores it. Using the Open Affect API we can get all the emotional states sent by a particular developer and manually extract his profile.

III. User Interface

In this project, our purpose was not to have a complicated interface but to have an easy and intuitive interface to help the user using it without any help and for us to easily test and debug our interface and application. The figure below shows the basic plugin interface.

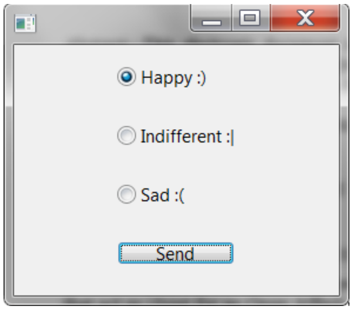


Figure 1. Plugin Graphic User Interface

This is a simple java plugin with as base the code given by a plugin code example from the Eclipse IDE. We have just put three radio buttons corresponding with an emotion (happy = 10, indifferent = 5 and sad = 0) and a button to send the current emotion. To communicate with the Open Affect Client, we have added a custom HTTP Client in our code to make an HTTP POST request to the server.

Only one parameter must be hard coded and it is the IP of the Open Affect Server in the HttpHelper.java file. The other parameters (or HTTP proprieties) automatically set are the HTTP method type (POST), the charset (UTF-8), the Accept type (application/json) and the Content-Type (application/json).

To send a correct JSON file accepted by the Open Affect Server we have been inspired by an existing example. We only change than the current timestamp of the request and the emotion related to the selected emotion on the plugin. At the end, the POST request is transmitted to the server at http://IP_SERVER:8080/api/measures

IV. SCRUM

Scrum is an iterative and incremental agile software development framework for managing product development. It defines "a flexible, holistic product development strategy where a development team works as a unit to reach a common goal", challenges assumptions of the "traditional, sequential approach" to product development, and enables teams to selforganize by encouraging physical co-location or close online collaboration of all team members, as well as daily face-to-face communication among all team members and disciplines involved. [1]

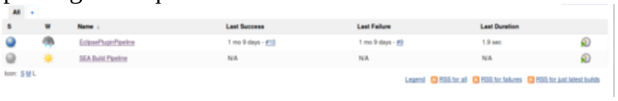
This method helped us to separate roles and to improve our product fastly. The product owner role was portrayed by Misha and the scrum master role by Elisa. The development team were the three of us. We passed through the distinct phases of SCRUM methodology to optimize our meeting's targets and minimize time loss.

V. Continuous Delivery pipeline

Jenkins is a software written in Java that permits to setup up a pipeline.

Jenkins helps to automate the non-human part of the whole software development process with now common things like continuous integration and by empowering teams to implement the technical aspects of continuous delivery. It is a server-based system running in a servlet container such as Apache Tomcat. Builds can be triggered by various means, for example by commit in a version control system, by scheduling via a cron-like mechanism and by requesting a specific build URL. It can also be triggered after the other builds in the queue have completed. [2]

A software pipeline is a sequence of actions and tests used to automatically build a release without developer's intervention. On failure, the pipeline shows where a test has failed and you need to restart the process from the beginning. Otherwise, on success, you have a build of your software. In our Jenkins pipeline, we take the latest version of our project on GitHub and compile it using a direct call to Java compiler and passing the required libraries to it.



All +

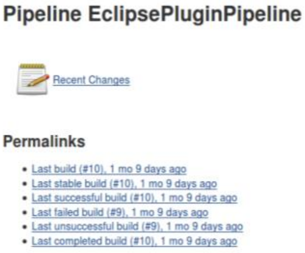
S	W	Name ↓	Last Success	Last Failure	Last Duration	
●		EclipsePluginPipeline	1 mo 9 days - #10	1 mo 9 days - #9	1.9 sec	
●	☀	SEA Build Pipeline	N/A	N/A	N/A	

Icon: S M L

Legend RSS for all RSS for failures RSS for just latest builds

Figure 2. List of Jenkins pipelines

Each pipeline has his own status where we can see if the last time the pipeline has failed the build or not. If we go into details and click on the pipeline item, we can control the pipeline. It's also possible to decide whatever the build is a stable build or just a development build.



Pipeline EclipsePluginPipeline

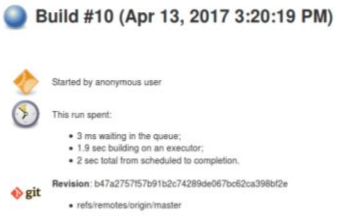
Recent Changes

Permalinks

- Last build (#10), 1 mo 9 days ago
- Last stable build (#10), 1 mo 9 days ago
- Last successful build (#10), 1 mo 9 days ago
- Last failed build (#9), 1 mo 9 days ago
- Last unsuccessful build (#9), 1 mo 9 days ago
- Last completed build (#10), 1 mo 9 days ago

Figure 3. Detail of our pipeline

If we click on each build attempt we can see how much time has been spent in each phase: waiting in the building queue, execution of the build commands and the total time. We can also see the ID of GitHub commit used to build the software.



Build #10 (Apr 13, 2017 3:20:19 PM)

Started by anonymous user

This run spent:

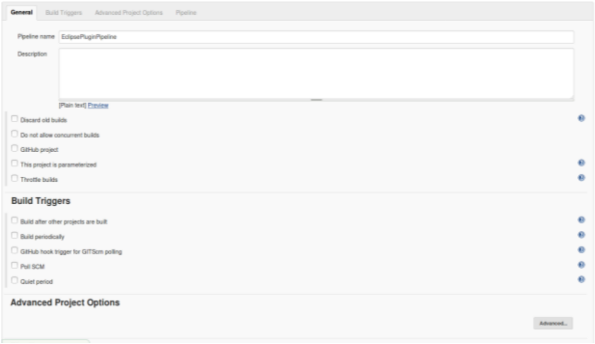
- 3 ms waiting in the queue;
- 1.9 sec building on an executor;
- 2 sec total from scheduled to completion.

git **Revision**: b47a2757f57b91b2c74289de067bc62ca398bf2e

- refs/remotes/origin/master

Figure 4. Detail of a build in our pipeline

In the Configure tab, we can adjust the pipeline script that build the software. Our script is divided into 2 sections or stages: Setup, Commit.



General | Build Triggers | Advanced Project Options | Pipeline

Pipeline name: EclipsePluginPipeline

Description:

[Plain text] Preview

□ Discard old builds
□ Do not allow concurrent builds
□ GitHub project
□ This project is parameterized
□ Throttle builds

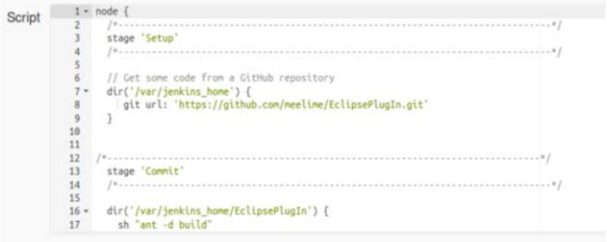
Build Triggers

□ Build after other projects are built
□ Build periodically
□ GitHub hook trigger for GITScm polling
□ Poll SCM
□ Quiet period

Advanced Project Options

Advanced...

Figure 5. Configuration options for a pipeline



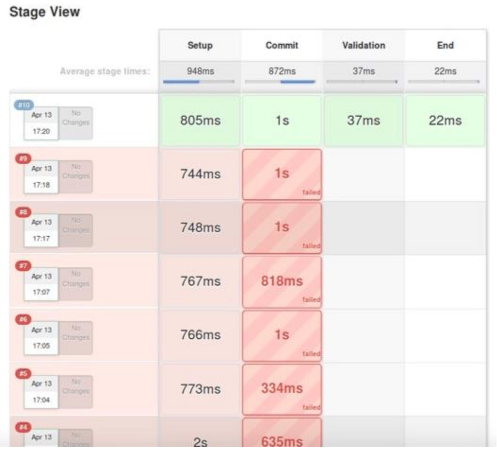
Script

```
node {
  /*-----------------------------------------------------------------------*/
  stage 'Setup'
  /*-----------------------------------------------------------------------*/

  // Get some code from a GitHub repository
  dir('/var/jenkins_home') {
    git url: 'https://github.com/meelime/EclipsePlugIn.git'
  }


/*-----------------------------------------------------------------------*/
  stage 'Commit'
  /*-----------------------------------------------------------------------*/

  dir('/var/jenkins_home/EclipsePlugIn') {
    sh "ant -d build"
```

Figure 6. Our script used in the pipeline

The Setup stage is where we define variables, URLs, and directories where the build will be placed. In our case, we define the GitHub URL where to fetch the latest version of our plugin and Jenkins use the GitHub URL to download a copy of the software in a specified directory. The Commit stage is where we compile the Eclipse plugin using Ant. At the end, we have a build of the plugin that has been built automatically using our continuous delivery pipeline.

When building a pipeline, it's also possible to see how many time was spent in each stage.



Stage View

	Setup	Commit	Validation	End
Average stage times:	948ms	872ms	37ms	22ms
#10 Apr 13 17:20 No Changes	805ms	1s	37ms	22ms
#9 Apr 13 17:18 No Changes	744ms	1s failed		
#8 Apr 13 17:17 No Changes	748ms	1s failed		
#7 Apr 13 17:07 No Changes	767ms	818ms failed		
#6 Apr 13 17:05 No Changes	766ms	1s failed		
#5 Apr 13 17:04 No Changes	773ms	334ms failed		
#4 Apr 13 No Changes	2s	635ms		

Figure 7. Detail of stages in our pipeline

VI. Results

We haven’t tested our emotional meter with real developers, but we have enjoyed the using of a continuous delivery pipeline and SCRUM method as they permitted us to make the work more fluid, easy and separate roles. We also think that this kind of emotional tests can really improve the job of developers because help to not waste mental resources when the developer is tired or unsatisfied. If the developer is always in a good and fit mental state, he can produce better and less error-prone code. The usage of a pipeline in software development helps to automatize boring and repetitive actions in compilation, and ease the identification of errors as each phase of the build is separated and isolated.

References

[1] Wikipedia, “Scrum (software development)” https://en.wikipedia.org/wiki/Scrum_(software_development), page 1, retreived the 20th May 2017.

[2] Wikipedia, “Jenkins (software)” https://en.wikipedia.org/wiki/Jenkins_(software), page 1, retreived the 21th May 2017.