Presentation:

Welcome to my internship presentation. This presentation contains information about the organization and the responsibilities performed throughout the period of six months between March and August. I will share with you some reflections based on my experiences during the internship from the perspective of an engineering student.

The first part of the presentation offers an overview of the organization including the company, the product and my team. Then I will talk about the context, the problems and tasks carried out during my internship. Following, some details of the projects and their respective analysis will be talked about. Finally, I will end with a few closing remarks.

IBM is Abbreviation for International Business Machines Cooperation. It’s founded in 1911 in New York. IBM produces and sells computer hardware and software and provides hosting and consulting services. the revenue of 2019 is about 77 billion dollars. IBM is one of the largest IT, infrastructure and IT consultancy companies in the world. Nowadays, IBMers continue to develop advanced technologies, such as Watson artificial intelligence, the Cloud Computing, and the Blockchain.

I worked in a team called ODM Evolution responsible for the product ODM abbreviation of Operational decision manager. It is a set of technologies and methods for clients to automate their day-to-day business decisions. Business decisions are everywhere in every organization in public and private sector. Tax Calculation, Benefit Calculation, Service Eligibility detection are examples of business decisions in the public sector. They change frequently for many different reasons like market needs regulation, competition and even customer expectations.

ODM Enables these companies to automate and manage these frequently changing business decisions. Business Team can be directly involved in management of rules by focusing on the business logic.

This scenario here is an online loan application powered by decision automation from IBM ODM. If a client sends a loan request from the application. At first eligibility is determined using some pre-defined business rules, if the loan is rejected,the result will be directly sent back. and if the loan is approved, it will be sent on for quoting。 So, this is the core business decision of the loan system. But the result of decision could be changed because the rules behind could be changed. IBM ODM allows business experts of the bank to easily update the rule-based system. Here’s an example of the a rule….. Because the rules could be defined in English, in French or some other natural languages, business users could manage the rules with little dependency on the IT department. So with this system, the business users are not required to have any programming knowledge, because they can use their natural language to implement the business logic.

Let’s dive deep into the product. Operational Decision Manager includes two main components: **Decision Center** and **Decision Server** .**Decision Center is for**  business departement. Rule authors policy manager can operate directly on rules without help of IT department. **Decision Server** is for developers to program to automate the response of decisions. **Rule Execution Server** provides the runtime environment for running and monitoring decision services. In this system, IT **users:** They develop and maintain the decision services. **Business users :**They develop and maintain the business logic. They can work in collaboration to build a decision management system. The collaboration allows them to focus on their own tasks.

The ODM Evolution Team is responsible for product maintenance, testing, building and delivery of ODM. It consists of a RES sub-team (responsible for Rule Execution Server module of ODM), a DC sub-team (responsible for Decision Center module), a QA & Software Factory sub-team (responsible for the product building and Quality Assurance) ... , I was attached to the RES team , responsible for developing tools to facilitate the debug process and to analyze the program more efficiently.

the **Continuous integration** (CI) is used to help us merge our code changes to a shared branch. The internal platform for us is called **Versatile Testing Tool (VTT)**, a Testing and Reporting system that can help us to execute the testing automation, and then report the test results in a web dashboard application. The testing automation is executed on the remote machine. Here’s an example of the report of an execution. We can see that there are some failures for the third, the forth and fifth classes so we know that we need to correct the error.

The problem is that We can see the failing tests, but we can’t easily figure out the reasons why they failed. It is possible that the difference of operating system makes the result of the same test different. In this case, it’s hard to locate the error because we can’t debug on our developing machine because we couldn’t locate the bug. That’s the first problem.

The second problem is that our product supports running on many different environments. On many different web servers and with different databases testing should run with the use of different browsers (Firefox, Chrome) or even different OS. In the development environment, it is difficult for us to execute testing with different combinations, but on the remote machine, we can easily run in all required combinations. So, solving these problems is the subject of my internship.

Through the first stage, with the help of the team, I quickly became familiar with the product ODM , the development method, the structure of ODM and the implementations of testing automation with selenium. The first stage built a foundation for my futures projects. and I also learned some communication skills though the weekly meetings .

After the first stage, there was a lockdown because of the coronavirus, and we all started to work from home. At the beginning, It was hard for me because the bandwidth of the Internet at home was not good so it was inconvenient to work. However, after communicating with the manager and mentor, the problem was quickly solved by a Wi-Fi amplifier. This experience taught me the importance of communication which I have always ignored before. From then on, I always communicated the progress with the mentor, And they can always give me advice and tell me what needs to be improved.

The idea is to fetch the release environment on local machine so that developers can reproduce the exact same errors. To implement this idea,

1. Work directly on the remote machine. Because we’ve installed all environments we need, we can use some remote-control software or ssh to connect to the remote machine.
2. We can use a virtual machine based on the Hypervisor technique. There isn’t a stable version of Vm software for MacOS. So, we chose to use docker.

The implementation of the first idea:

The goal is to move the source code to a remote machine, configure the development environment and construct the project structure.

Given that we already have the Linux system with the test environment, all we need is to download the source code by using scp and to install our development tools. And then use ssh to connect to the remote machine and begin your work.

The remote machine will be reset every several days. To avoid installing and configuring the env every time we lock a remote machine, we decided to automate the process by using a shell script. Developers can directly develop and test in the released environment after launching this script.

Although this project is simple, there are a few things topay attention to. First of all, since we only need src to be copied to the remote machine, log files and dependencies should be filtered out. Then, to simplify the usage, the script will install all the dependencies if they haven’t been installed yet. Also, there is a configuration file (format ***ini***) in which we can easily define the URL, user, password and destination of the developing tool of the remote machine.

Through the above design, users only need to run the script to build the environmenteasily.

This is a screenshot for the internal repository of github. I developed twoshell scripts, one to install eclipseon the remote machine and the other to read the configuration from config, build the project and copy the source filesto theremote machine. And I also wrote a documentation to explain how to use these scripts and how it works. These scripts also help Fengyi who hopes to run eclipse on the remote machine.

This is a solution: We move the debugging process from the local machine to the remote machine. The development environment is exactly the same as the VTT release environment. The problems are solved.

But the disadvantage is also obvious: when we don’t have a high bandwidth, we can only use **ssh** to connect to the remote machine, which is not convenient. More importantly, there is a limited number of available remote machines. It happens that we won’t be able to use a remote machine when the platform is busy. But we wish to continue our work even when we don’t have a remote machine!

The second idea is to Simulate the release environment on local machine.

Virtual machine and Docker are two different options in this case. Given that there isn’t a stable version of Virtual Machine Software for MacOS. We decided to build our own Docker image.

Docker uses OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through channels.

All containers are run by a single operating system kernel. It runs from an image. We can start , stop and restart the container. If we want to save the instance of the container , we can commit and we get a new image. We can build a docker image from .

**Dockerfile** or pull pre-defined images from the docker registry or load the image from a compressed file. So the idea is to run the docker container based on Linux OS, on my computer which is based on MacOs.

To construct and execute the correct docker container , we need to:

1. Pull the Red Hat image. There is already a Red Hat image on Docker hub, so we can use it directly.
2. Installation of necessary developing tools such as **wget** **unzip**, IBM JDK, Eclipse... in Docker.
3. Configure the system environment in Docker.
4. Build the Docker image, share files with the host .
5. Execute a Docker container using the Docker image and configure the X protocol (to display GUI applications).

Steps 1,2,3 are encapsulated in the Docker file, and steps 4,5 are encapsulated in a script.

Here’s a screenshot of the documentation of this project from GitHub. Developers only need to launch 2 scipts to run the docker container. The first is to …. We only need to launch the script called config to the second…. We need to execute the script called launch to achieve this goal

Through this project, we directly built a virtual environment on the local machine that is almost the same as the release environment, and it is convenient to use. Because of the binding technology, the source files in the Docker is directly shared with the local machine: Local modifications are visible to the Docker container, and the modifications inside the Docker container is also visible to the host machine. We don’t need to worry about the transmission of the files.

This implementation of the project works well, but nevertheless suffers from a serious limitation : we can’t execute multiple tests in parallel on the same server. To support such parallel test process executions, I refactored the project and changed it into a server-client architecture. The advantage of the separation of the two container instances is that we can run multiple tests on a server, and we make the container more lightweight. But we need to think about how different containers communicate with each other.

### Network In Docker

The **bridge** network is useful in this case. The bridge driver creates a private network that is internal to the host so containers on this network can communicate with each other. Here, we created a network name ‘mybridge ’ which allows those 3 containers to communicate with each other. And Of course we can launch more than 2 clients at the same time .

Through this project, we can directly build a virtual environment on the local machine that is almost the same as the **release** environment. It is quite simple to use and we don’t need a remote machine. So, it’s better solution if developers are familiar with docker.

This internship was very fruitful to me because I had to cover many different fields. In terms of technical skills, I learnt programming in Java, developing testing automation with Selenium and scripting with shell.

I had a chance to experience the deployment of a web project with several web servers on the Linux server, and the technics of virtualization. On the other hand, I also experienced the whole software development process, which taught me to follow some structures (Client-Server architecture, separation of configuration and execution) to make the code readable and extensible. In terms of management, I experienced the Agile which makes flexible responses to change.

The real teamwork unlike the college project where one person did all the work and the grades were shared among the rest. Here, it is more professional and each one in a team needs to work together to finish the task. We need to always work as a team without focusing entirely on ourselves because we can’t work efficiently without the help of teammates. And I leant the some communication skills like how to talk concisely and confidently.

In a word, it was a valuable experience because I gained professional expertise for my CS career

Slow a little bit.

+ France Lab,

+ 2 Part of test

+ Plus, de words in ppt

+ Une slide sur le résultat ( avant - après )

+ Docker, testing on local machine, ( for QA ) update easily

25 min 44 .