Popular science summary of master thesis work at Industrial Electrical Engineering and Automation, LTH, Lund University.

AUTO-TESTING OF CODE

Thesis worker: Flamur Breznica

Testing in the automation production system is usually done manually and physically, what if it was possible to do the test automatically with the help of a computer instead. Are there enough tools to do this and can the process be more time-saving? This report will aim to answer those questions.

The thesis primary subject was to investigate if there are any possibilities to automate the tests needed on automation production systems or at least cover a part of the tests automatically. Is it possible to do it and to what extent? All automatic things need work for it to be automatic, nothing is free and hard work will be needed for making something automatic, no matter what someone must put in the work. This thesis was built up by one strategy and one software design pattern which is commonly used in the software world, the software design pattern is called Model-View-Control and the strategy is called the test automation pyramid. Model-View-Control is usually used in software and is divided into three parts. The view is what the user sees, while the model decides what happens behind the scene and lastly control says what it wishes is going to happen, then the model decides if it is possible or what will happen. [1]

The test pyramid is an old and rare strategy that describes how the testing on software should be performed. By dividing the testing into layers in a shape of a pyramid, it will be easier to separate the test in different layers and how much time and focus should be spent on each layer where further down of the pyramid equals more testing. Let’s use a calculator as an example, the numbers in the calculator correspond to units. The numbers in a calculator are the foundation of the device if the numbers in the calculator do not work properly, the calculator is faulty. Which means the biggest focus and more testing should be done here. The next layer would be the service test, in this case corresponds to adding, subtracting or dividing. Lastly the top layer, which is the user interface, how the calculator should look like. [2] Figure 1 illustrates how the pyramid is interpreted.

Since the software production differs from the automation production system an interpretation is needed to be done, or else it will not work, and the testing will only be done on half of the system since automation production systems have both hardware and a software side that communicate. By making tests automatic, it will prevent some of the faults to come up since it will start at the beginning. First thing is to control the input if they are correct. By doing this you control the source, a common saying is that the result will never be better than the source is since they depend on each other.

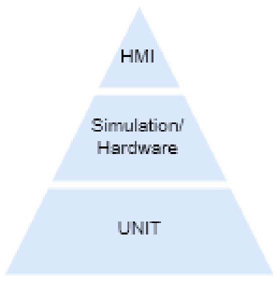

Figure 1: Illustrates how the interpretation of the test pyramid looks like.

Testing machines are not an easy task, unlike apps or software on phones and computer, machine code is written on a computer which needs to communicate with hardware, for example a motor. Usually this is done by a controller that handles the logic for it to make sure it works as planned. However, since the hardware is absent and the test is done automatically with software, there is no insurance that the test made sure everything turns out as planned, thus automatic testing can lead to new problems when compared to software. The best way to deal with this is to try it on real hardware, to make this work takes practice. In apps or programs there are great ways of making the testing automatic but on the automation production system the evolution has been that it is too expensive and just not worth it. The need for testing and finding errors early has always been there, even before creating a product it would be the first thought. The thesis aims to investigate these problems and implement them on a virtual machine that was also created in this thesis.

As the pyramid test shows, the unit testing has great importance, see figure 1. To test the units, a controller was needed. This was programmed and to run the tests a controller was simulated. The controller that in theory will control the machine during the process. Due to shipment troubles, this was done manually in the beginning and later automatically. The next step was to create a view, which from the pyramid strategy was interpreted as HMI and according to Mode-View-Control strategy, the view is one of the three parts that are important to implement. This was done before simulation to give a visualization of the virtual machine and have an overview of the process. The last step was to create the model which would interpret as a simulation and with these steps, a virtual machine was created.

To test the units, it is needed to use tools since these tools are not implemented in the environment the code is written in, which is commonly integrated with the case of software programming. By using a program called "S7 Unit Test", the tools will be implemented and thus the pyramid will also be implemented and automatic. The unit tests are done automatic and it will also test the code automatically after the code has been written. This leads to an accurate and quick response given through both in the program and an automatic report. However, the first tests were done manually due to the delivery problem which later became automatic.

Automatic testing is coming to the automation production systems and is most likely here to stay. It probably will not save time, but it will move time earlier in commissioning and the quality of the product may be of a higher standard. Unit testing as earlier described is a program that does everything by itself. It will have a learning curve, but it will become better and more accurate with time, as the developer gets used to it. Implementing a test took about six hours while running a test with the tool took 36 seconds. This is about the same estimated time needed when parts of the tests were done manually in the beginning. However, if the manual testing needs to be redone, the automatic testing will most likely more time-efficient.

Currently the old traditional testing is done by the physical machine which leads to a lot of traveling between different machines on updates. The companies would like to lower the travel distance as much as possible while raising the quality of testing. This does not mean that automatic testing will replace the traditional testing but some of them might be replaced and yield better results.

The need is here, but the publisher should be more interested in making automatic testing as a part of their software instead of giving it as a standalone program. This is probably one main issue with why automatic testing is not used.

Working with automatic testing and simulation will lower the risks of ruining actual machines since this is done virtually and is cheaper. If a problem is found earlier then it can be fixed earlier which could prevent future implications. The faults that are discovered might have been discovered out on site and the costs would have been higher as well. Automatic tests and simulations are possibly reusable for machines. The first machine that is created is going to cost more but will have a learning curve since it can be reused and improved which could save expenses later. Timesaving will become more and more, evaluating automatic testing is very hard since it depends on case to case. But it is most likely to move time in the beginning and possibly save it in the future. Instead of performing tests manually, the time must be put on creating the testing cases. However, if they are re-used then it could save time and eventually raise the quality of the products

The simulation of the thesis could be improved since it is done with old traditional software while the testing is done with a new tool. The new software SIMIT is a new way of handling the problems with simulation, it gives both a better overview and will test the finished controller software without modifications.

According to Nikolova, the cost will rise as you go higher on the pyramid. [3] To save time with automatic testing is all about prioritizing and choose wisely. To automate unit testing is usually profitable and according to Nikolova, 90 percent of the unit testing should be automatically. There is always a need for testing, even though the tests are done automatic it does not mean the tests will find and remove all faults. Automated testing will remove some errors and create others, it will have an expensive start cost and be hard in the beginning. But the need for testing will remain, no matter what is done. With the tools that exist today, new doors are opened. It is possible to pitch ideas and sell them to customers if they need convincing, or raise the bar for the product quality, or move the need of having physical testing machines to needing servers instead. The hunger for better machines is always there, and the automated test is possibly a step in that direction.

[1] G Krasner, S Pope, *"A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System"* 1988. Available: https://www.researchgate.net/profile/Stephen_Pope/publication/248825145_A_cookbook_for_using_the_model_-_view_controller_user_interface_paradigm_in_Smalltalk_-_80/links/5436c5f30cf2643ab9888926/A-cookbook-for-using-the-model-view-controller-user-interface-paradigm-in-Smalltalk-80.pdf [Accessed: 2019-12-10]

[2] M Cohn, “The Forgotten Layer of the Test Automation Pyramid” 2009. Available: {https://www.mountaingoatsoftware.com/blog/the-forgotten-layer-of-the-test-automation-pyramid [Accessed: 2019-12-16]

[3] Z Nikolova,Testing Strategies in an Agile Context2020.Available :32 https://link.springer.com/content/pdf/10.1007%2F978-3-030-29509-7_9.pdf[Accessed: 2019-12-17]

Master thesis work 2020: ***Auto-testing of code****.*

Supervisors: Mats Lilja, Christian Isaksson.