Continuous Integration
How it helps to efficiently automate an Automotive Toolchain

MAGNA Telemotive GmbH and Nippon Seiki (Europe) BV

By Maximilian Kaltenhauser, Core Developer Telemotive Test Automation
and Rebekka Haisch, Product Manager Telemotive Test Automation

Abstract

In this whitepaper, a continuous integration toolchain for use in the automotive sector is demonstrated. The basis of this proposal is the cooperation with Nippon Seiki Europe where this toolchain was successfully implemented and used for automated HMI testing. The device under test in this specific case was the prototype of a Head-Up Display produced by a premium German automotive brand.

Keywords: Continuous Integration (CI), Jenkins, Telemotive Test Automation (TTA), Hardware in the Loop (HIL), Software in the Loop (SIL), Version Control (VC)

Content

Introduction

With the rise of Industry 4.0, automation also found its way into the automotive industry. Not only are the cars manufactured and assembled with the help of robots, also testing of e.g. display functionalities or other E/E functions is now possible with software; often referred to as "Software in the Loop" testing. Before, the bottleneck in manual testing at the Tier1s was expanding with a strongly increasing complexity and number of test runs demanded by the OEM, increasing labor and tooling costs and a stagnating number of available testers on the labor market. With test automation, reliable tests running 24/7 are possible. Nippon Seiki, a renowned supplier of instrument clusters for automobiles and motorcycles, has undergone this process of shifting from manual to automated HMI (Human Machine Interface) testing with continuous integration. With Telemotive Test Automation, the entire toolchain of Nippon Seiki has been automated, which lead to a strong decrease in testing effort. In the following, the process of efficiently automating an automotive toolchain with the help of continuous integration is described.

Continuous Integration

First, let us take a look at what continuous integration means and how it can aid an automated testing process to achieve a seamless transition from automated tests to a reporting interface. "Continuous Integration (CI) is the process of automating the build and testing of code every time a team member commits changes to a version control. CI encourages developers to share their code and unit tests by merging their changes into a shared version control repository after every small task completion. Committing code triggers an automated build system to grab the latest code from the shared repository and to build, test, and validate the full master branch (also known as the trunk or main)." [1]

In automotive terms, the toolchain is either triggered via a software update provided by the Original Equipment Manufacturer (OEM), or when the test scripts are being adjusted. Alternatively, it is recommended to run certain regression tests at least once per release or once a week to be on the safe side. At Nippon Seiki, tests are executed every night. The amount of existing test cases comprises almost one week of non-stop testing, which makes it impossible to test each of them every night. Therefore, a prioritization of test cases is done and depending on their priority, the test cases are executed more or less frequently. After the Continuous Integration toolchain is triggered, the goal is to require minimal user interaction to run the tests and finally present the results in a human readable form that can quickly be analyzed and communicated to all stakeholders.

The need for a Test Automation Framework

To seamlessly integrate into common continuous integration toolchains, a test automation framework is required that supports established software such as Jenkins [2] or Version Control Systems like GIT or Subversion. These requirements are met by the testing framework TTA (Telemotive Test Automation), which provides additional functionalities needed in the automotive sector.

Telemotive Test Automation is a dedicated automation tool that enables flexible adaptation to any hardware or software. Its modular structure is shown below in Figure 1. Based on Python, the tool enables the developers to use open source packages and to benefit from elaborate reporting with clear visualization. Pre-assembled units, code examples and easy-to-apply test case implementation enable a straightforward learning process for new users as well as faster development cycles. Telemotive Test Automation is continuously improved at the pulse of the market and can be customized for individual requirements. [3]

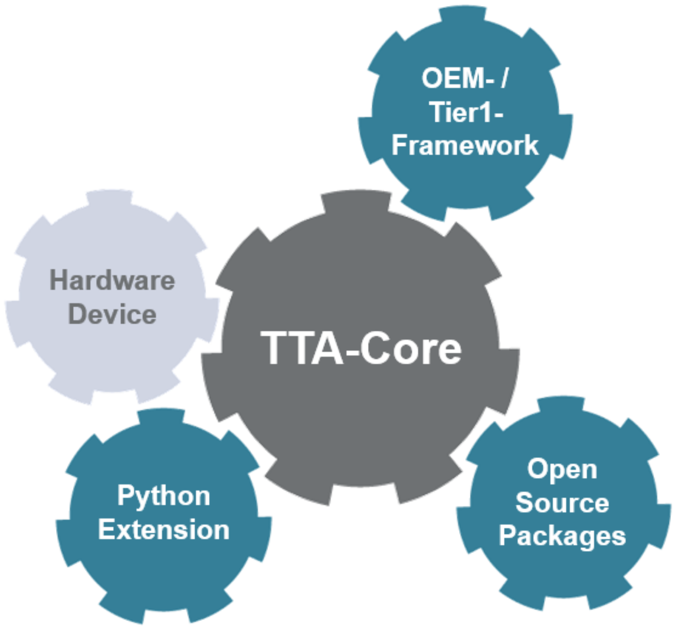

Figure 1: TTA modular structure

In the continuous integration process outlined in this whitepaper, MAGNA Telemotive offers a way to bring different tools together and to oversee the testing process whilst providing clear feedback and user-readable results at the end of the test runs. To achieve this, a custom solution of TTA has been offered to Nippon Seiki. Existing implementations were improved alongside, guided by their valuable feedback.

Overview of the Integration

In Figure 2, the deployment model is shown as it is used in the cooperation with Nippon Seiki. Not represented in this graphic is the automated export of the created results into a web-based test management tool and the cross-references to the entire TTA reports on the Jenkins server to streamline the evaluation process.

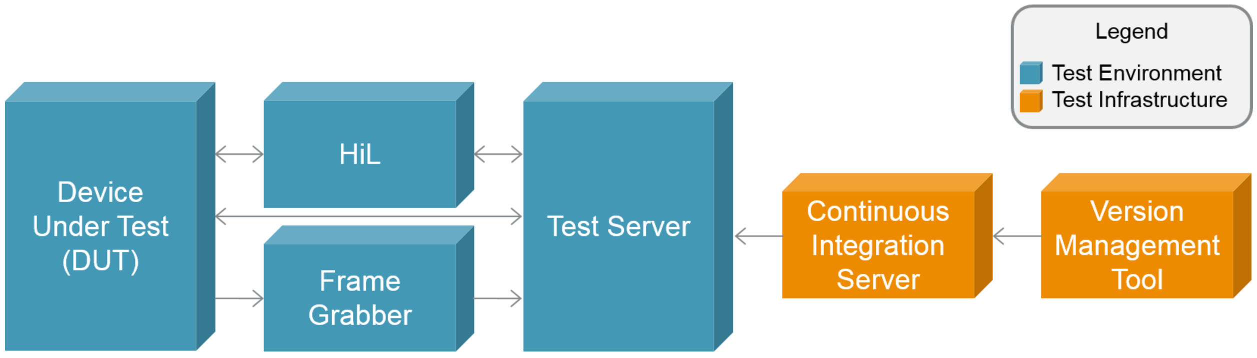

Figure 2: Deployment Model

This figure is split into two major parts, the test infrastructure and test environment. The infrastructure comprises anything that is part of the CI process (or triggers it) but does not interface with the hardware (device under test). The test infrastructure describes the Jenkins job in further detail and visualizes any interaction with the hardware such as simulating the BUS traffic or catching images via a frame grabber.

For similar procedures with a different device, a full simulation (here Hardware in the Loop) or a frame grabber might not be required; additional hardware can be added when necessary. The general process remains the same regardless of the specific test environment. The process is triggered by the test infrastructure (specifically the CI Server), which starts the test server (TTA) to interface with the device under test (DUT) and perform the tests.

Hardware in the Loop

In order to provide the needed test environment to successfully test the DUT, all incoming signals have to be simulated on the BUS systems. This can be achieved via a Restbussimulation (RBS) [4], either directly in TTA or via a third party software. In the project with Nippon Seiki, the OEM already provided a functional RBS. Here, TTA is able to interact with the third party development and testing software tool through the COM (Component Object Model) interface and adjust the simulation at runtime, to fit the project’s needs.

On the hardware side, the DUT is connected via a CAN device to the test bench. Moreover, a DVI (digital visual interface) based frame grabber is used to capture images that can later be used in test cases for comparison with MAGNA Telemotive's reference library.

Version Control

Nippon Seiki uses the Version Control System Subversion as a standard for projects in this division. For similar use cases, Git would also be an option. The task to version the test cases and trigger the CI process can be accomplished by both systems.

Jenkins Integration of TTA

As seen in Figure 3 below, TTA is part of the Test Server section of the deployment model and therefore responsible for the execution and reporting of the test cases itself. Here, both have a direct connection to the DUT, as well as the ability to control the Restbussimulation side and to interact with other hardware devices - in this test environment a frame grabber.

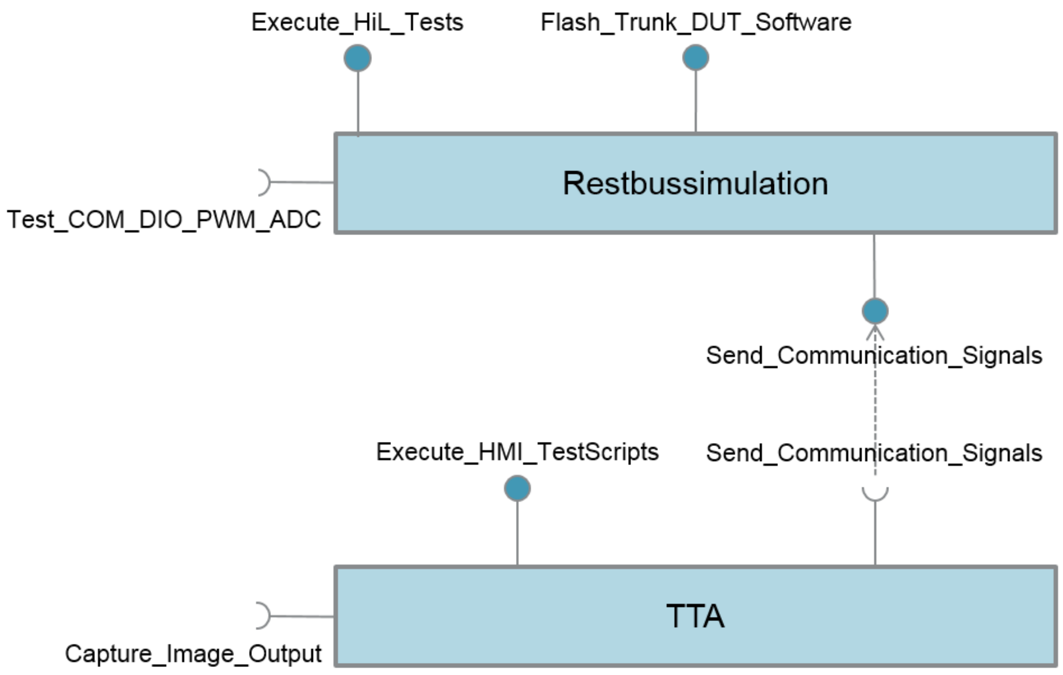

Figure 3: Test Server

A Restbussimulation is generally used to simulate the expected environment so that the DUT can function just as it would if all the sensors and other components were actually connected to it. There are multiple tools that can generate this expected traffic on the BUS, for the described project a Restbussimulation based on a third party development and testing software tool was provided. Alternatively, this can be achieved directly through TTA itself. Hence, the third party software tool can be started over a COM interface, a specific configuration loaded and parameters or trigger functions adjusted in real-time.

Apart from handling the RBS, TTA also interacts with the frame grabber in order to be able to receive pictures from the DUT and compare them via image processing and comparison algorithms. Here, the received image is compared with the library of reference images that was provided by the OEM, to uncover discrepancies and mark the test steps accordingly. To limit the amount of test cases that need to be created, optional test case parameters are used so that one test case can be used for a multitude of reference images (for example checking against different full color images to see if colors are displayed accordingly).

In the next step, TTA is able to create test sequences automatically by running a custom script, which collects all the reference images and fills in the parameters for the test cases individually. Alternatively, this step can be achieved by manually creating the sequences in the TTA Graphical User Interface (GUI) and exporting them as JSON files. These sequences can then be loaded by TTA and executed on the Jenkins server by directly calling the TTA Runner via the command line instead of loading the GUI.

Reporting

For reporting, a web-based test management tool is used, which provides a known environment for both the testers/developers and the test managers. The latter must communicate the results to the OEM and analyze bugs that might be found in the current prototype. The scalability and accessibility of the used test management tool are other advantages, which make it the preferred choice.

In order to generate generic XML reports that can be imported into Jira, several options are provided (alongside TTA’s more detailed HTML based reporting structure). For future projects, the TTA developers are currently working on a Jira REST API [5] integration that will ultimately aim to make custom importing scripts obsolete and fully support Jira based test management.

Conclusion

Today it is commonly known that there is a crucial need for test automation within the software industry, since the amount of data to be processed is steadily growing, there exist various possibilities on how to process that data and the increasing amount of interfaces exceed the capabilities of manual testing. Thus, more and more companies start to integrate automated test management into their strategy.

The process described in this whitepaper is not only used by Nippon Seiki, but also in the Telemotive Ethernet Testhouse and continues to be improved and adapted. The goal is to automate this testing process as much as possible in order to provide an easy and straightforward process for test verification in the automotive industry. Once the here described toolchain is set up, the DUT can be exchanged but the connections as described in Figure 2 remain the same, leading to a high reusability and a further minimization of effort. Enabling this reusability is a big advantage of TTA. In the future, the need for improved automated test case creation will further increase. Together with Nippon Seiki, the developers of Telemotive Test Automation will continuously deliver innovative solutions.

About the Authors:

Maximilian Kaltenhauser is a Core Developer of Telemotive Test Automation and a Python expert at MAGNA Telemotive. He has been working in various test automation projects and shares his experience in trainings and consulting services.

Rebekka Haisch is the Product Manager of MAGNA Telemotive’s business area Software Solutions. She is responsible for Telemotive Test Automation, among other products, and strives to digitalize companies, e.g. with automated instead of manual testing.

References

1. Sam Guckenheimer. What is continuous integration — Microsoft, 2017. [Online; accessed 4-November-2018].
2. Shwetha Sneha Kunja. What is jenkins? — Vmoksha Group, 2018. [Online; accessed 4-November-2018].
3. Magna Telemotive GmbH. Telemotive test automation — Magna Telemotive GmbH, 2018. [Online; accessed 4-November-2018].
4. National Instruments. The fundamentals of restbus simulation — National Instruments, 2015. [Online; accessed 4-November-2018].
5. Atlassian. Rest apis — Atlassian, 2018. [Online; accessed 4-November-2018

MAGNA

DRIVING **EXCELLENCE.**
INSPIRING **INNOVATION.**