How Do Manufacturers of Embedded Software Debugging and Test Tools Test Their Products?

For safety-relevant applications, tool manufacturers have to subject their software tools to rigorous testing. The experience gained in this process can be for the benefit of customers as well.

Summary

Chapter 8-11 of the ISO 26262 standard, a version of IEC 61508 specifically "adapted" to automotive, is dedicated to confidence in the use of software tools. This standard has a specific impact not only on vehicle manufacturers and their suppliers but also on the development and test processes of tool manufacturers and thus the development process of the actual software tool.

This article depicts the organizational and technical measures iSYSTEM has derived from current functional safety standards to support automotive software development departments in the preparatory stage and especially during the so-called tool qualification phase. Experience from earlier tool qualifications in aviation (DO-178B) have been incorporated in this process.

The measures discussed have a sustainable impact on the development and test process of a software tool manufacturer to the effect that processes are reassessed and optimized, thus further enhancing the outward transparency of these processes. This degree of transparency is accomplished most efficiently based on test automation and is not only for the benefit of the automotive industry but for the benefit of all customers.

The starting point for high-quality software and hardware are internal process definitions that precisely describe how these products have to be developed and tested. Test of Embedded Systems means test of hardware and software at the same time.

Optical test systems and boundary scan tests have been used in the hardware sector for many years, and so-called dynamic tests have been added. These advanced tests verify different types of memory (Flash, RAM, …), memory access time and peripherals in test mode. The required speed is achieved by using the microcontroller on the test system. iSYSTEM also share their know-how with partner companies in this context.

Software testing focuses on the generation of new software, archiving of old and testing of specific software versions as well as an operational version control system. Continuous new and further developed microcontroller architectures as well as extended functionality of an embedded software application cause an increase in software complexity. Consequently, test automation is required. Another motivation for test automation is the demand for intense testing in the scope of functional safety standards. Tool manufacturers thus create a foundation of trust in their products. This article is dedicated to the topic test automation.

Test Automation @ iSYSTEM

iSYSTEM internalizes test automation with fitIDEA, a test tool suite that facilitates fully automatic execution of defined tests in embedded systems.

fitIDEA is an automated test tool that was initially developed for iSYSTEM internal purposes. It uses a generic API (application programming interface) of the iSYSTEM development environment named winIDEA (isystem.connect) and the public scripting language Python.

As a tool manufacturer iSYSTEM faces almost the same challenges as their customers: Lots of different hardware variants, historically grown code base plus legacy code, and extended functionality according to higher requirements of customers, increase the complexity of development and test tools as well and makes it evident to introduce test automation.

Image 1: Tool test bench for automated regression tests executed on different hardware platforms and target systems. The three test benches also include customer-specific systems that contain up to 24 complete systems each (blue box and target system).

All critical aspects of the embedded software debugger and integrated unit test tool suite are tested. Amongst others, the following functions are checked:

- Standard debugging and IDE functions like download to flash memory, target system memory access (read, write, real-time access), breakpoint functions (execution and access)
- Specific time measurements to test advanced debugging, like trace and profiler functionality
- Software test with code coverage and unit test
- API functionality

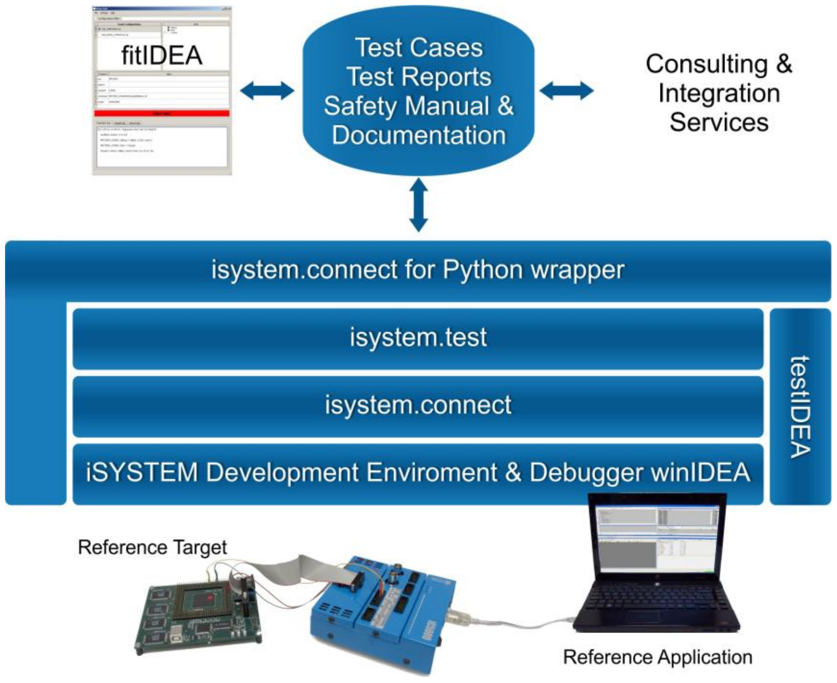

Image 2: Software architecture for regression tests

The fitIDEA concept is based on the deterministic behavior of the test environment comprising the following components:

- Reference system (standard board of the semiconductor manufacturer, iSYSTEM board, customer-specific system)
- Reference application for such a board/system
- iSYSTEM debugging hardware (blue box)
- iSYSTEM software (winIDEA and testIDEA on PC side)
- Compiler

The reference application for the respective target system is the key element. It is programmed thus that a deterministic (defined and reproducible) behavior is exhibited:

Example:

```
char g_c;
void Increment()
{
  g_c++;
}
```

The execution of this function obviously increments the variable g_c by one. fitIDEA now carries out the following steps:

- Execute the code until Increment() is requested
- Read the value of variable g_c
- Execute the function through to the end
- Read the value of variable g_c

The test was passed if the second read-out of variable g_c shows that the value was incremented by one. This test covers the following functions:

- Identification of code and data symbols from the download file (set breakpoints to symbolic values, read variable values through symbolic names)
- Interrupt of execution (a breakpoint was set on function entry and exit)
- Execution of code and CPU status verification (run until, wait until CPU stops)
- Read access to memory (read-out of variable values)
- C compiler: code and debug information was generated

Even if a component of the test environment changes (iSYSTEM tool, compiler, target, ...), the application has to maintain its behavior, and fitIDEA, as the test tool, still has to deliver the same result. The Python test scripts are created by dedicated test engineers. The iSYSTEM reference application uses reference commands **(//rt)** to save expected results of a test. These references are stored in the source code of the references application (see image 2) and are thus independent of the compiler used.

```
char g_c;
void Increment()
{
  g_c++; //rt step_and_evaluate: value[+1]
}
```

So far, more than 60 hardware configurations in the tests benches (see image 1) are actively running tests, and each configuration has about 100 test cases. Each developer can access all configurations and can also run tests locally before taking over the modified program code. Automatic testing with fitIDEA is executed overnight, and test results are available the next morning, thus facilitating seamless integration. New test cases are continuously added based on new functionalities, customer applications or resolved support cases.

What have we accomplished so far?

- New, verified software can be generated more quickly
- More reliable software traceability and documentation
- More peace of mind for CTOs and CEOs
- Test process transparency for customers

What can customers accomplish with iSYSTEM?

Customers can use iSYSTEM's fitIDEA in their company. When using fitIDEA as execution platform, customers also get a reference board and reference application with the related test scripts for the specific microcontroller.

With the support of iSYSTEM, this reference design and the related test scripts are now ported stepwise to the customer's hardware and software. The tests are extended and adapted as required.

Eventually, the customer has a complete test environment for his hardware and application software that facilitates continuous, efficient and quick testing of all components. If a component is exchanged or modified, e.g. a new compiler or hardware version, testing is implemented easily and variations are identified immediately.

Outlook

Interestingly, functional safety standards more and more incorporate the qualification of software tools. This is of particular interest because the risk of using software tools in respect to specific requirements and "use cases" has to be considered in preparation of a development. Software tool manufacturers are not excluded either, i.e. these manufacturers are expected to implement suitable improvements and extensions in their own development and test processes. iSYSTEM have already aligned their processes to this trend and internalize an agile software development process with test automation transparency.

"In a way, all test methods are interesting. However, the methodology and the company have to match up." – Erol Simsek iSYSTEM AG

Integration of test tools in the test process has to be easy: *Interview with Erol Simsek, CEO of iSYSTEM AG, about the necessity of testing in embedded environments and the integration of tests in the development process.*

What challenges to development and test engineers does the tool suite fitIDEA overcome?
As a start, fitIDEA is the internal implementation of iSYSTEM's software development and test process. Our focus is mainly on agile development and an internalized process of continuous integration. We have therefore approached the issue test automation systematically. fitIDEA is moreover an environment to prove the correctness of the iSYSTEM tool functionality according to certain use cases, mainly customer-specific ones. This is a requirement of several functional safety standards.

What are the characteristics of an efficient test tool?
This is hard to generalize. After all, there is a wide variety of tools that cover many different techniques – from modeling tools to static code analysis, unit/integration test tools and, last but not least, system test tools. Most of them are a combination of software and hardware. For sure, a key aspect is that the tool is easy to use and can be integrated in the process easily. This creates acceptance among development and test engineers as well as managers.

How can a test technique master the complexity of modern software applications and the variety of devices?
The answer is fitIDEA and internalized test automation in an agile software development environment – in other words, automation and sophisticated techniques. And an occasional glimpse at the aviation sector with its extensive experience in model based/requirements based testing.

There are diverse approaches and philosophies regarding software testing. Which one do you think is most interesting?
In a way, they are all interesting. However, the methodology and the company have to match up. At this time, I think agile software development and the related testing philosophy are quite fascinating. Especially because many habitual processes and patterns are breaking up, and any type of software challenge is overcome with both determination and professionalism.

Some experts expect static code analysis tools to become more significant in future. Will this diminish the significance of testing?
That's right. The significance and deployment of such tools has increased substantially over the past decade and is just gaining momentum. Basically, such tools are easy to integrate in existing development and test processes without major changes to the actual process. Static code analysis tools cover part of the test process and are a useful addition to increased test depth.