JOCHEN WENDEBAUM

GO ONE BETTER

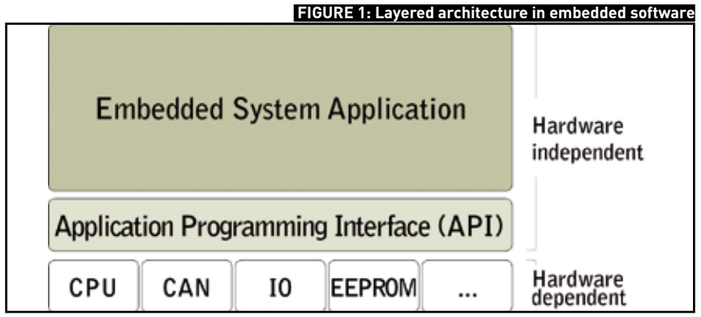
A NEW SIMULATION-BASED TEST STRATEGY FOR CONTROL SYSTEM SOFTWARE CAN PROVIDE A TEST ENVIRONMENT WITH HIGHER OBSERVABILITY, MORE CONTROLLABILITY AND GREATER INTERACTION POSSIBILITIES

Testing is one of the single most important disciplines to improve in the embedded software industry today, if it is to ensure the production of high-quality products along with the ever-increasing demands for more functionality. Testing is the primary method to guarantee that a system works according to its specification, to locate problems and deviations, and to detect architectural mismatches in system integration. However, software testing in the industry is, in general, immature. A multitude of test methods may be applied, testing is often not well planned, and as a result there is a trade-off between quality and far more expensive projects than planned.

One of the problems is that the engineers performing the tests are limited to working with the target hardware for testing very basic software functions. The access to this hardware is often limited, implying that it may take a while before an implemented function can be tested. When the target hardware becomes available, each compilation, code-loading and booting sequence takes some time to complete before the test can be initiated.

Furthermore, the observability of the application in the embedded computer is often not as good as the observability of software executing on a PC. Engineers report having to perform basic software tests at the sites where the machines are used, and needing to invent ad hoc test methods to be confident that a specific deeply embedded software function behaves as intended. In such cases the machines may only be available outside normal working hours, leading to an exhausting schedule and implying high risk and costs.

FIGURE 1: **Layered architecture in embedded software**



CC Systems has found that at least 85% of the faults found during testing can be far more efficiently detected and corrected in a simulated environment on the developer's host PC. Most of the faults are actually related to basic systematic errors in the coding or thinking process by the designers and programmers, or to architectural mismatches when software modules are integrated, such as resolution of data, or erroneous expectations of poorly specified modules. These problems are not dependent on the compiler or the hardware platform and should be tested in the most efficient environment for that purpose, which is the environment provided by modern development tools on the developer's host PC.

If a developer is testing on a standard PC, the test environment is only a mouse click away, anywhere, at any time – decreasing the time between implementation and feedback. The test can be started without performing any code loading or rebooting, it can be far more controlled, and it can be observed using modern state-of-the-art development tools for PC applications. CC Systems believes it should be industry standard to test basic algorithms and software-to-software integration directly on the developer's host PC.

With this in mind, CCSimTech simulation technology enables the code for an embedded control system to be tested directly on the developer's host PC. The target hardware can be exchanged and used in later stages when hardware-software integration can be in focus. Automated testing and fault injection, which might not even be possible to do on real target hardware, can also be performed easily in this simulated environment.

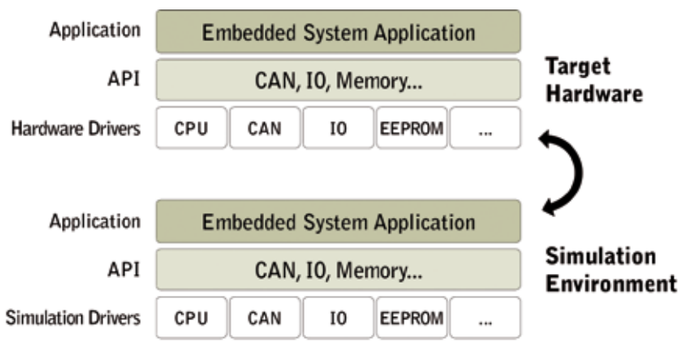


FIGURE 2: **Comparison of the target and the simulated environment**

Technical concept

CCSimTech's technical concept (Figure 1) is based on software usage of device-drivers to read and write inputs and outputs, to access the internal memory and for communication, via CAN or LAN for example. The application itself 'talks' to a well-defined interface, the hardware abstraction layer (HAL) which itself delivers all the data to and from the hardware.

Figure 2 visualises the core concept in the simulation environment, where the drivers are exchanged, and the new drivers don't access real hardware but instead talk to a virtual software layer. The application itself is exactly the same as on the target system, using the very same HAL, except the drivers behind enable the running of the software on a standard PC. Those simulated drivers are the core of CCSimTech.

When the system is ported for execution on top of virtual device drivers, a complete system including all system nodes can be executed and tested on a standard PC, without the need for expensive hardware. Data exchange occurs on virtual buses rather than physical buses, and sensor and actuator interaction is performed through virtual I/O channels.

Models of the client's machine can be deployed, and relatively cheap and far more efficient tools from the PC software-development community can be used, such as the automated testing tools, debuggers and performance profiling tools shown in Figure 3.

CCSimTech also supports hardware-in-the-loop (HIL) tests, i.e. parts of the system can run on real target hardware. This is made possible through seamless incorporation of PC adaptors for communication, such as CAN-based protocols, and I/O cards for PC that are capable of performing real digital, analogue and PWM I/O operations against the target hardware. In this way it is possible to test a complete system, even if some parts of the system are black-box third-party components delivered as pre-programmed hardware. This also enables moving gradually towards incorporating more and more of the real target hardware in the later development stages when hardware/software integration must be performed.

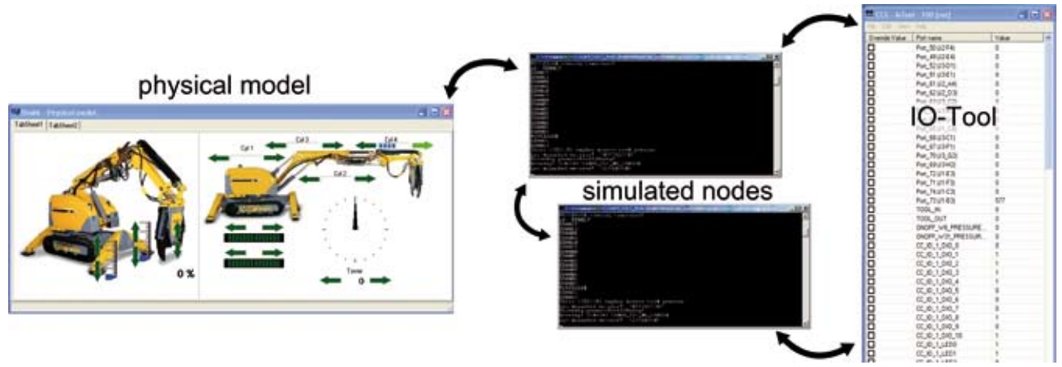


FIGURE 3: **Complete simulation in one PC showing, left to right, machine model, the control system nodes, and a signal-logger tool**

The overall benefits

Having observed the application of CCSimTech simulation-based testing internally and with several customers using the technology in in-house projects, CC Systems has identified several benefits.

Many software test activities can be moved to earlier phases – tests can start even before the target hardware becomes available. This means that costly bugs that are usually found late in the project can be eliminated earlier in the development cycle.

The software also gets more total test time, even if the development project can be shorter. Each test case can be performed more efficiently directly on the developer's host PC, and testing is more parallel as there is no need for serialising access to the target hardware.

A simulation-based strategy opens up test cases that are otherwise not possible, or are problematic to carry out, on the target hardware. This is because of the highly increased interaction possibilities when applications are run on PCs rather than by remote testing the target hardware.

Software integration on the target hardware can be performed with confidence that the software logics adheres to the specification. Remaining problems are efficiently isolated through applying the heuristics on the basis that they most probably stem from hardware/software integration issues.

No short cuts

A machine's quality is increasingly dependent on the quality of its software. There is no known shortcut to ensuring software quality – each part of the source code must be extensively analysed and tested. It is therefore, of course, prudent to provide the most efficient test environment the developer needs at each development stage. For a clear majority of the source code's testing needs, the most efficient environment is the developer's own PC. This is a test environment that enables higher observability, controllability, and interaction possibilities and can easily be created by deploying a simulation-based test strategy. **iVT**

Dr. Jochen Wendebaum is a system engineer at CC Systems AB

CONTACT
www.cc-systems.com
info@cc-systems.com