Migrating from debugging to testing embedded software

Cordemans Piet, Boydens Jeroen

May 8, 2010

KHBO Dept Industrial Sciences & Technology, Zeedijk 101, B-8400 Oostende, Belgium

{Piet.Cordemans, Jeroen.Boydens}@khbo.be

Abstract

Testing embedded software anno 2010 is mostly limited to ad hoc debugging, only focusing on the current issue. However as the complexity of embedded systems steadily increases, rigid testing of software components will be inevitable. Fully fledged test suites, running automatically, have proven their value at testing business application software, as manual testing is expensive and error-prone. Furthermore Boehm's law determines that the later a bug is discovered the more it will cost to deal with the problem. Testing the software early on in the development cycle, will be more profitable. The extreme case, which embodies these ideas, is a software methodology called Test-Driven Development (TDD). Test-Driven Development advocates writing a test prior to the business logic, defining the intended behavior. This leads to an incremental development of both the test suite and the business logic, targeting test coverage of more than 80%.

However applying these principles in embedded software development is not straightforward. The limited memory footprint and cross-platform development issues are impediments typically posed by embedded hardware. In the worst case the hardware is only partially or totally unavailable at some point in the embedded software development phase.

To deal with these issues, an embedded development process can start with the development of hardware independent functionality, gradually adding more hardware specific features. In first instance these features need to be supported by software stubs and later on verifying the stubs on the real implementation. Based on this idea, three coping strategies have been defined. These strategies are categorized according to the place of execution of respectively tests and code. First, both tests and code are executed in a host system. Next, the tests still reside in the host, but the code has been migrated towards the target system. Finally, both tests and code are executed on the target embedded system. These strategies embody the trade-off that needs to be made between testing accurateness and speed of test implementation and execution. Current tools for embedded debugging, such as an instruction set simulator and logic simulation, can be used in the context of developing or testing on a host system.

The results of the ongoing research are presented, as these principles are demonstrated in an academic context. Further research includes proof of concept applications in an industrial context and specific measurements of test coverage in the different strategies. Furthermore, embedded and testing issues should be addressed. These issues encompass testing of real-time behavior and non-deterministic characteristics, such as concurrency.