



A reliable development process strengthens links between product development and global suppliers.

By **Mark Underseth**.

Forging links

Today, embedded software development involves a supply chain of engineers, offshore outsourcers, open source and third party software vendors; all delivering hundreds of software components.

In a recurring pattern, software development appears to progress smoothly to the implementation phase. From that point, however, projects enter long and unpredictable integration and system test phases. This indicates companies are struggling to integrate and verify so much software from so many sources. As a short term fix, managers add more engineers and resources, with limited effectiveness and high cost. Software releases are still often delivered late with compromised quality.

It's no surprise that embedded software is getting more complex. For example, we've gone from a few thousand lines of code in mobile phones 10 years ago to millions of lines today, running on full fledged 32bit multitasking embedded operating systems with full memory protection. Yet most developers are, essentially, still using the methods of 10 years ago.

With software content in electronic products doubling every two years, manufacturers are finding they can no longer go it alone. To keep up, they employ more engineering resources, license more third party software, leverage open source if pos-

sible and outsource more development.

Of course, it is not enough just to create the software; we need to verify that it operates correctly on increasingly complex hardware. Unfortunately, embedded software verification methodologies and practices haven't changed significantly over the last two hardware generations.

At one time, single board computers had only a general purpose cpu. They then upgraded to SoCs or network processors and now multicore processors. During this evolution, the corresponding embedded software has increased exponentially. The problem is that software testing processes haven't evolved to manage the new scale and complexity of embedded software development effectively.

Integration and testing already consume disproportionate amounts of development time and resources. Compared to software implementation, it is not unusual for integration and system test to take up to 10 times as long and to be staffed with large teams. Plus, too many defects are escaping into the late stages of development, wreaking operational havoc on software teams whilst making software delivery agonisingly unpredictable.

Without achieving the fundamentals of a scalable, reliable and predictable software process, verification and integration challenges will only worsen. For example, how

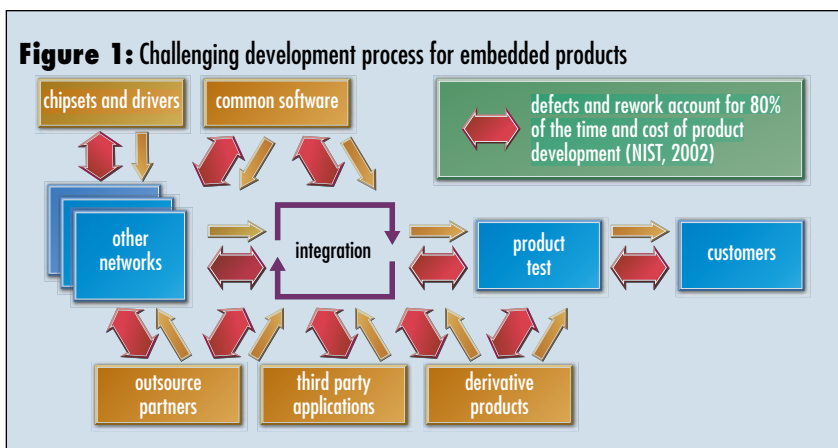
might embedded software development be impacted by multicore processors? Multicore processors might force development teams to be more disciplined in their overall software design. In addition, specific types of software will be executed on certain purpose built cores. As a result, more cleanly architected software design and code might be forced to be more 'componentised', with well defined interfaces between major functional areas.

Still, software continues to be an enterprise scale effort. Even if the volume of code needed to enable and deliver the various features was essentially the same, there would be an additional dimension of software complexity from multithreading or concurrent threading. Ultimately, the various software components would still need to be integrated, tested and verified to operate properly as a whole. With current methods for integrating and testing complex embedded software already breaking down, increasing levels of software complexity will exacerbate the problem.

Changing the practice

A new model is required to optimise the delivery of complex embedded software. The key objectives of this new model would be to validate software components earlier; reuse and integrate code more effectively; automate more test processes;





and increase visibility into software quality earlier in the development cycle.

Using supply chain management principles, it is our contention that all software suppliers – internal and external – should create and deliver test assets. These fully automated and reusable tests would have lasting value because they could be used throughout the development life cycle, in derivative projects and by other functional teams to integrate, triage and verify software. Test assets also provide detailed and ongoing visibility into the health of the software deliverables.

Creating and sharing test assets effectively requires a unified verification framework that supports collaboration with suppliers, enables better leverage of domain expertise and which facilitates automated testing and early verification strategies. The verification framework provides the common infrastructure to reuse, automate and execute effectively tests from all developers, and support test management and reporting. It must also facilitate different types of testing, such as unit level and API level testing. Further, the verification framework must scale beyond a single

engineering group and be able to aggregate test assets from internal engineering teams and external software suppliers.

Cultural change

This new approach requires cultural change. Engineering teams (suppliers) have to change the way they approach embedded software development by treating applications – even individual software components – as products in and of themselves. In essence, developers need to ask what they can do to maximise the quality, reusability, portability and ease of integration of the software being supplied.

Engineers tend to create tests with no expectation of them being reused by other teams or processes. Developers' test code is generally 'throwaway', rarely reused by others and not automated easily as part of a test portfolio. With a unified verification approach, engineers should consider how a part needs to be tested and integrated upfront. Importantly, developers should create tests that are reusable, automated and have lasting value.

To begin implementing this new verification strategy, start with the most impor-

tant suppliers – the internal development teams. Starting with one functional team, developers can work together define the tools, coding standards, test requirements, test asset management, gating conditions, policies and other aspects of a common test framework and process.

Measures of success

Much of the success or failure of processes is dependent upon the culture of the engineering organisation and no single prescription works for all. Only by measuring can you identify what can be improved. Look for quantifiable factors that have a large impact on the ability to deliver high quality software on time.

Once internal processes are producing satisfactory results, align external suppliers. By requiring suppliers to use the same verification platform, standards and processes, they can deliver reusable test assets that can be automated along with your own. This enables your engineering teams to accept and integrate the deliverables and updates of software suppliers, because they would be using the same automated tools and techniques as the internal engineering team.

Unifying the verification strategy for the software supply chain within your engineering organisation, as well as aligning external suppliers, is a large undertaking. However, the benefits are substantial. Organisations would eliminate most manual testing, automate integration, support multiple product releases efficiently, optimise software reuse and manage their software suppliers effectively. 🔄

Author profile:

Mark Underseith is founder and cto of S2 Technologies.

