The role of DevOps in electronics design and test

Turn the pages of most IT publications right now and it's hard to avoid the term 'DevOps'. An increasing number of firms in the electronics and semiconductor industries are also beginning to adopt this software development methodology. In essence, it's about bridging the gap between the development and operations teams, enabling them to work together in a more collaborative, transparent and seamless way. This helps bring products to market more rapidly, without compromising quality. ***Robert Cowham, Senior Consultant at Perforce Software****, explores why DevOps matters for electronics design engineers and how to ensure successful adoption of the methodology.*

DevOps involves thinking beyond just managing code to cover all digital assets (for example, component designs, CAD drawings and prototype information) in a unified, continuous pipeline that enables automation and testing at every stage of a digital asset's lifecycle. It's also a natural partner to other methodologies, including Agile Development, Continuous Integration and Continuous Delivery.

With organisations under pressure to deliver shorter, faster and more frequent product release cycles – together with greater volumes and complexity of digital files – the need to efficiently manage end-to-end software lifecycles has never been greater.

Without a unified view of each stage, compliance becomes difficult and time-consuming to verify, projects can go off track, and digital IP (intellectual property) becomes more vulnerable to security risks.

With organisations under pressure to deliver shorter, faster and more frequent product release cycles, the need to efficiently manage the end-to-end software lifecycle has never been greater.

The secret to DevOps success

Of course, any methodology is only as good as its execution, and many firms are still at early stages in their DevOps adoption – with plenty of lessons to be learned along the way. Fortunately, there are some strong DevOps pioneers out there, who have trodden a trail from which other firms can benefit.

A single source of truth

A 'single source of truth' tracking every change is critical for efficient DevOps. It should be able to cover all assets, with a single view across all contributors, and a traceable and documented history of the project at every stage (including who worked on what, when and how, as well as an instant view of the current status).

In the electronics market, this can be a challenge, given the way that many designers work. For example, the binary file assets being created by analogue designers do not lend themselves intuitively to a digital 'single source of truth'. Similarly, embedded software managers might have code, library and object files, spread across different platforms, each with their individual configurations.

So, the 'single source of truth' needs to be able to support all contributors, regardless of

their existing workflows, preferred tools or systems. Once this is in place, it then becomes easier to deal with issues such as defects or deviation from project requirements, as well as give external auditors the data required for compliance processes. The 'single source of truth' also fits well with Component-Based Development (CBD), which requires all the digital components associated with a product to be centrally stored and tracked, using efficient meta-tagging to make future location and re-use easier.

Version everything

The typical basis for a 'single source of truth' is a version control system – but again, 'buyer beware': not all systems are the same, and this may make it impossible to achieve 'true' DevOps. Look for systems that offer the ability to support multiple file formats, including binary content, and which work around users without dictating a new set of processes (we all know that engineers will find a workaround if they are not happy).

Also, look for a version control system that can scale quickly to support escalating volumes of complex files; offer support for remote or external and distributed work teams; and provide clean integrations with other systems in the toolchain. Finally, ensure that the version control system provides an immutable history of events (in other words, one that ensures the facts cannot be modified at a later stage).

End-to-end

Software development is just one step in successful DevOps adoption. For it to work well, everything needs to be covered: ideation (such as feature requests); definition; design (including requirements specification); development; testing; deployment; release; and maintenance. This end-to-end view makes it easier to ensure that nothing is accepted that deviates from the 'single source of truth' (or if it does, it is easier to spot). This could include a developer being asked to work on something that was not specified in the original set of requirements.

Testing should also be automated, as far as possible, with rapid feedback to the development team to ensure issues are identified and addressed as early in the process as possible. Automation supports predictability and repeatability, enabling speed and volume at scale.

DevOps in action: u-blox

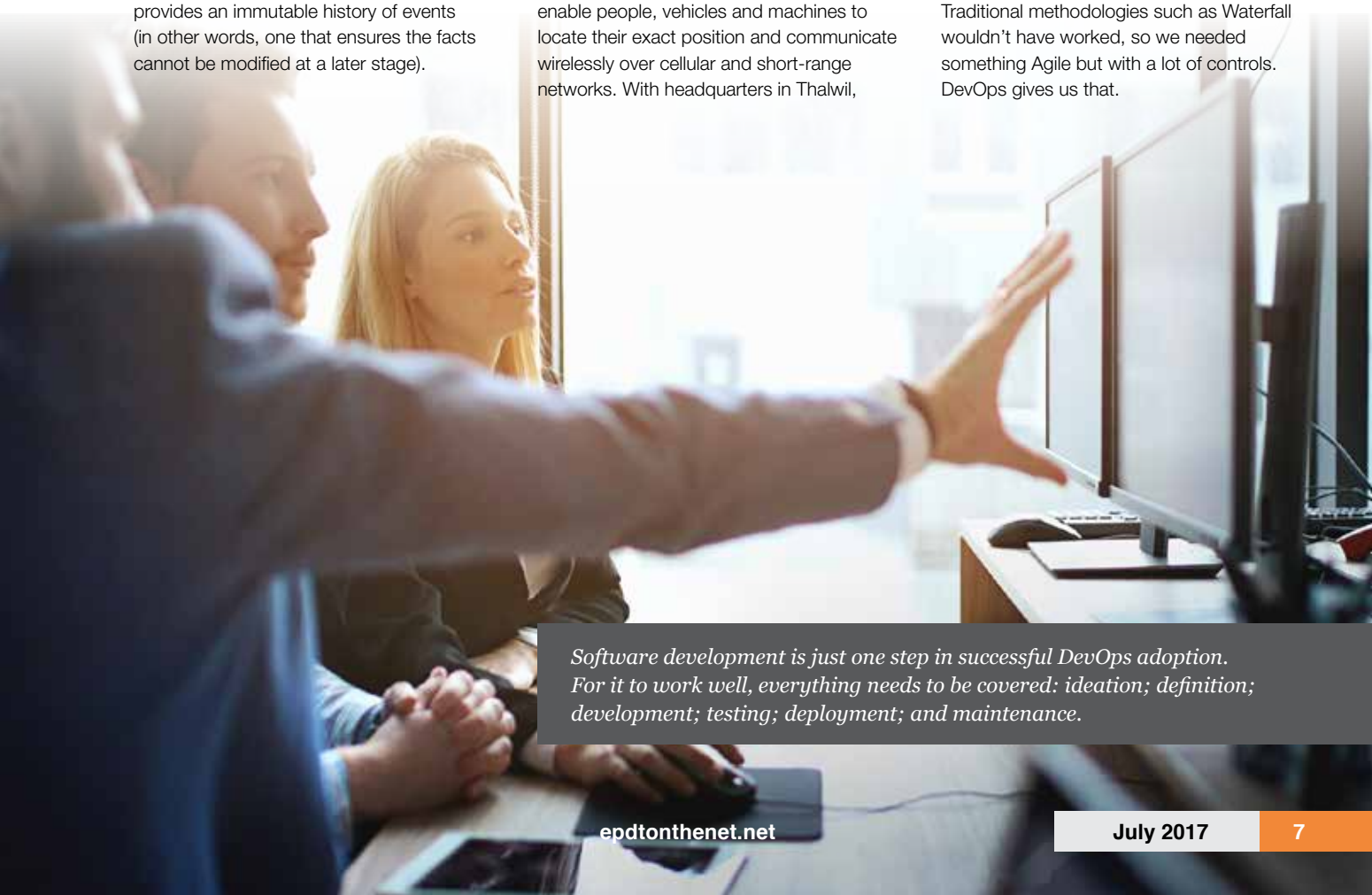
One example of an electronics design organisation that is successfully implementing DevOps is Swiss-based company u-blox (recently shortlisted for the Computing DevOps Awards). u-blox is a global leader in wireless and positioning semiconductors and modules for the automotive, industrial and consumer markets. u-blox solutions enable people, vehicles and machines to locate their exact position and communicate wirelessly over cellular and short-range networks. With headquarters in Thalwil, Switzerland, u-blox also has offices throughout Europe, Asia and the US.

It has chosen to adopt DevOps to ensure successful execution of one of its most ambitious projects in many years: the introduction of its new LTE Cat 1 wireless modem, which is part of its growing portfolio of products and technologies aimed at applications for devices connected to the Internet of Things. This is also a largely software-defined modem, which is in itself evolutionary.

Reason for DevOps

u-blox decided to use DevOps for what is probably the largest embedded software development project in its recent history because of the particular challenges it presented. Firstly, there are hundreds of contributors, across multiple teams in multiple locations around the world, all operating in different time zones.

Secondly, as Stephan Uyttebroeck, Principal Software Engineer, based in u-blox's Leuven location in Belgium explains: "Due to the time pressures and scale of the project, we had to make a start as soon as possible, even before there were clear software and hardware specifications. We knew we would have to start testing and building software, even if things changed along the way. Traditional methodologies such as Waterfall wouldn't have worked, so we needed something Agile but with a lot of controls. DevOps gives us that.



Software development is just one step in successful DevOps adoption. For it to work well, everything needs to be covered: ideation; definition; development; testing; deployment; and maintenance.

"DevOps ultimately supports faster time-to-market. For instance, it reduces the time between a bug being identified and being fixed. Things get sent back fewer times, because there are more reviews and tests between the development and production teams along the way – long before there is a final release version. Our pipeline is largely automated, but there are some manual tests as well."

DevOps progress

u-blox reports that this first implementation of DevOps has worked well, and attributes this to a couple of factors. Firstly, it was applied to a new project and environment, rather than teams being asked to change habitual workflows applied to existing or previous projects. As Uyttebroeck says, "This was such an important project and we had a great deal of support from company leadership, so people have been motivated to make it a success."

Secondly, u-blox's carefully thought-out strategy to support DevOps, through both process and tool adoption, has made a significant contribution towards successful implementation. "Rather than enforcing particular tools," Uyttebroeck explains, "we gave people guidelines, best practices and tried to lead by example – but we also provided them with flexibility and responsibility."

Providing users with flexibility

"For instance, we implemented Helix from Perforce Software as the preferred version control engine, among other tools such as Jenkins for builds, JIRA for bug-tracking, Nexus for artefacts and Swarm for code reviews. If the team find transition to a new tool is too much for them, then we allow them to take a step back and agree reduced level of interfacing and delivery – as long as they take responsibility for ensuring what they deliver works with what other teams are delivering."

In practice, the majority of users chose the recommended tools in the first place, and even the ones who initially stayed with the status quo have made the transition since. Says Uyttebroeck, "For example, one team wanted to carry on using their existing version control engine, but when it saw that teams using Helix were working much faster, they made the move to Helix, because of the stability and performance in the backend that it provides. Plus, looking ahead, there is its ability to co-exist with Git." u-blox also gives developers the choice of using streams for pre-defined workflows, but again, this is not mandated.

The project is still in the throes of completion, but as Uyttebroeck says, "Although there are lessons along the way, things we have changed and other things we would like to change, we would not have been able to achieve so much and so quickly without DevOps. We are already taking some of the best practices from DevOps and replicating those in other projects. And for our big project, it has reduced the 'big bang' effect, enabling us to bring everyone together early and start working together, even before everything had been defined."

u-blox is a good example of an organisation that appreciates and has embraced the cultural challenges around DevOps adoption. While DevOps needs the right use of the right tools, it ultimately hinges on bringing together people and processes, plus explaining why these changes are happening – demonstrating how they benefit both individuals and the organisation. Putting in place a solid foundation that combines processes, cultural understanding, and tools that underpin the methodology, is the secret to a successful DevOps journey.

Rather than enforcing particular tools, we gave people guidelines, best practices and tried to lead by example – but we also provided them with flexibility and responsibility.