The Perfect Project

WNDRVR

EXECUTIVE SUMMARY

How software is developed and delivered affects the bottom line at most industry companies. Software now represents the bigger value contribution, and offers the possibility to differentiate your product from the competition. So how software gets developed, and how it gets delivered, are no longer isolated questions for the engineering department—they are executive staff questions that affect the company's bottom line.

Software development practices are undergoing dramatic changes. Continuous and Agile practices are used everywhere, all for the better. But inefficiencies are still huge in the embedded technology industry, and changes are slow.

It doesn't have to be this way. Using Wind River® Simics® for system simulation allows access to target systems, tools for collaboration across teams, and new possibilities for automating what is not feasible in the "physical" world—all of which means that Simics enables the fundamentals of Agile practices. In this paper we will detail how providing access, collaboration, and automation have achieved the following:

Cut debug time by **35%**

Saved **thousands** of man hours in development

Shortened bring-up time from **22** weeks to **24** days

Helped companies get to market **20%** faster

TABLE OF CONTENTS

THE PERFECT PROJECT

Your company builds complex electronic systems (networking equipment, systems for the Internet of Things, automotive systems, industrial robots, flight engines, etc.). In these systems, hardware meets software; your technical world involves multi-core hardware with multi-staged pipelines, heterogeneous systems, hardware accelerators, special purpose devices, ASICs, and more.

Your software ranges from advanced firmware and boot software to complex OS set-ups, virtualization technology, complex I/O, networking management, data throughput analysis, fault software diagnostics … the list goes on. Your world is complex.

You're also part of an ecosystem, internally, externally, or both. You rely on other departments, units, subsidiaries, or third parties to deliver to you on time, and they rely on you. You also need to enable your users—partners or customers—with your technology and software development kits (SDKs) for developing certain functions of the complete system. With this web of dependencies comes risk.

To manage these two factors, technical complexity and risk, you must address developer efficiency and risk management. If you succeed, your reward will be shorter time-to-market, shorter timeto-money, and lower total cost of ownership: the perfect project.

What Is Perfection?

Perfection is an aspirational notion, so really this paper attempts to describe a more perfect project. By piecing together real-life scenarios from actual customers into one composite project, this paper is intended to help you visualize what a better, more efficient future might look like.

We discuss five main activities: design, development, testing, integration, and enabling customers and your ecosystem. Within this framework, we give examples of how Simics has been used by our existing customers, in order to illustrate what a "perfect project" might look like. We also describe the significant impact simulation can have on developer efficiency and risk management. We'll leave it to you to consider how pieces of this perfect project could expand across your organization to become a portfolio of "perfect projects."

> Simics enables Agile and Continuous Integration practices by providing access, automation, and collaboration.

This paper is not about one project that accomplished everything discussed, but rather a composite of project components. Each individual component story is taken directly from one of our customers and is based on actual use. Every component, and every benefit discussed, has been experienced by at least one of our customers.

PROJECT PHASES

Design

Define Your System Before Hardware Is Locked Down

Every project starts with a design, or a definition. In the perfect project, using a virtual platform instead of physical hardware in this early phase enables you to define the system virtually and decide on a system design before hardware design is locked down and taped out.

If you have a blank slate, with no legacy to consider, a virtual platform allows you to be creative without limits. Of course, in the real world this rarely happens. Instead you likely have a previous generation of your product that you will reuse pieces of—either software or hardware, or both. Whether you are starting from scratch or building on a legacy, when defining your next-generation product you will have access to a virtual platform of the hardware, and you can now play with what-if scenarios:

What happens if you move a software function from device A to device B? What happens if you scale up memory to be very large? What happens if you run the software on 12 cores instead of two? What happens if you move connector type T from board A to board B, or if you switch to using connector type S? In the perfect project you will develop hardware and software in tandem. You will let software design influence the overall system design—not hardware design. The benefit of defining your system virtually in this way is that you try out scenarios before you lock down the hardware design and before you tape out. The value of this risk mitigation is nearly impossible to quantify because you really don't know what you don't know. But most would agree that hardware errors, late tape outs, and early design misses can be extremely costly.

Develop

Start Software Development Before Hardware Arrives

By using a virtual platform instead of physical hardware for development of your low-level software, you can get access to this platform four to six months before the physical hardware arrives.

> You will let software design influence the overall system design— not hardware design.

Generally this early access is the most obvious benefit that Simics customers experience. During the platform development phase, your engineers will be using the simulated version of the hardware (the virtual platform), but the code they produce for it will be the real production code—the same binaries that will go into the final product. When the physical hardware arrives and it's time for bring-up, the platform code is moved over to the physical hardware and works with few adjustments. We have customers who shorten bring-up time from 22 weeks to 29 days by using simulation. Some of our customers succeeded with bring-up in only two hours. One customer used to go into the bring-up integration phase with hundreds of bugs, and since they began using Simics, it's down to under 20. It's safe to say that using virtual platforms can dramatically reduce hardware bring-up time. Furthermore, since the code is executed on a simulator during development in the perfect project, your engineers have access to unparalleled debugging techniques. A simulator gives you complete control and perfect inspectability of your target, and will save your engineers 35% of traditional debugging time. The more complex the target is, and the closer the software is to the hardware, the higher the value of using the debugging techniques a simulator can offer.

One customer solved a problem in 30 minutes that would normally have taken a team of three people working three weeks to solve.

> We have customers who shorten bring-up time from 22 weeks to 29 days by using simulation.

SHARING THE VIRTUAL SYSTEM

Cross-Functional Teams: Fused Development, Testing, and Integration

Now you have a virtual platform of your entire system. The platform code is available, and the virtual replica of your system is up and running. This executable virtual platform can now easily be shared among the entire team.

In the perfect project, you don't separate roles from functions, which just creates artificial barriers and prevents efficient collaboration. Instead of "silos," you organize as cross-functional teams, according to system expertise. These cross-functional teams will be up to you to define, as best serves your business needs: network throughput optimization, fault diagnostic software, control software, customer installation and configuration … the possibilities are endless.

This approach saves time because you work more efficiently. How much time depends on the size of your organization and how far you can take the vision of "fused development, testing, and integration." One Simics customer has saved over 20,000 man hours … and counting.

Test

Parallelize and Scale

In the perfect project, you scale out testing to a degree that is nearly impossible using only physical hardware. By parallelizing test runs on an infinite amount of virtual hardware, you gain both time and quality. You automatically configure and re-configure your entire hardware setup in no time because it is done via scripts.

One customer configures complex labs in minutes instead of weeks. The team runs hundreds of different software workloads every hour. They load a completely new software stack for each workload run—and they do it for multiple configurations of their system in parallel. In addition, they connect network traffic generators to the system to test various scenarios. There is no practical alternative to implementing this practice with physical hardware, because having infinite hardware is simply impossible—too large and too costly.

We also have a customer who saved $3.2M per test cycle in man hours, just by letting the traditional test team use a simulation-based test bench as a complement to physical hardware. They simply augmented physical hardware with virtual hardware so that more tests could be run in parallel.

Integrate

From Continuous Integration to Continuous Deployment

What does it mean when you have cross-functional teams, working in rapid, small cycles, doing fast and continuous development and integration, having free access to virtual labs, with largescale automated and parallel testing? It means you can now move to continuous delivery or deployment. Deliver new features, capabilities, fixes, and quality to your customers in rapid cycles, and thereby accelerate business. If your business has anything to do with the Internet of Things, this capability may be a critical component for success. To deliver continuously, you need to get

your build cycles down significantly, and of course you will need to build and test on the physical hardware in the last stages. But to efficiently implement the rapid development cycles needed for continuous delivery, you cannot rely solely on physical hardware, which would simply be too costly and impractical. In the end, what continuous delivery or deployment means is increased quality delivered continuously to your customers, and being able to keep up with market demand.

Being able to implement continuous delivery has become a key competitive advantage for one of our larger customers, and is rapidly becoming important to more.

Enable

Support Your Customers and Ecosystem

It's time for your project to enable a customer or a partner to begin developing their pieces of the overall system. Of course they also want to get an early start, so now you deliver the relevant subsystem, both the software load and the relevant underlying virtual hardware, as soon as it's ready. Your customer or partner starts their own development based on the system you sent them, and you have made it possible for your customer or partner to shorten their time-to-market. Remember when you used to wait until after the hardware was past the alpha stage? Seems crazy now, doesn't it?

SIMULATION-BASED VIRTUAL LABS

In the perfect project you set up a virtual lab based on simulation. The virtual lab provides unlimited access to all sorts of virtual hardware platforms, which meansyou have all configurations set up simultaneously. You use the virtuallab for two main purposes:

1. Continuous integration for alldevelopers; fused development and testing
2. Automated, parallel, and largescale regression testing

Using a virtual lab has two main benefits:

1. Flexibility, which leads to acceleration: All individual engineers have access to the virtual lab, which can easily be scaled out to any size. Engineers use it for continuous integration and nightly build and test reports. Each team sets up the reporting as they prefer: logging, profiling data, test results … on a virtual platform everything can be controlled, and you can access any data.
2. Cost savings: Using a simulation-based virtual lab instead of a physical lab saves you floor space, energy, hardware costs, and so on. One customer saves 38 times the floor space of physical labs by using simulation-based virtual labs, and 15 times the yearly cost of using physical hardware by using virtual hardware. Anothercustomer saves $6M in target hardware lab costs.

Furthermore, if you have an SDK that you ship with your product, you now include the simulation tools in that SDK so that customers can take advantage of all the benefits you have.

We have customers who run their development in programs spanning three or four separate companies. They all enable one another by using simulation. And some of them save as much as 20% in time-to-market by collaborating with simulation technology.

Another aspect of enabling customers and partners is training—and training on a virtual version of your end product makes things more practical. Since it's all just software, you can send, share, and equip your students easily; all you need is a laptop. At Wind River, our VxWorks® training classes are performed on a virtual target, saving us time and making it easier to work with our students.

Now that your product is in your customers' hands and they've been trained, what happens when they need help? Support is easy when your customers are able to email their system configurations with a checkpoint that your support team can easily replicate. At Wind River we use the checkpoint functionality to mark and share a specific point within a complete system state. Our support engineers say, "Checkpoints are great to replicate the customer issue. It significantly downsizes the effort and time needed to duplicate customers' problems." When our support team can use a checkpoint to address a customer issue, it saves them 20%–40% in time.

the system you sent them, and you have made it possible for your customer or partner to shorten their time-to-market. Remember when you used to wait until after the hardware was past the alpha stage? Seems crazy now, doesn't it?

> The end result, simply put, is that our customers can now produce better software faster.

CONCLUSION

We've given you a look into the "perfect project" by using real examples of how Simics has been used by multiple customers. Each customer had a specific problem they wanted to solve with Simics, and most of them also discovered that Simics could solve problems they didn't realize they had. In most cases they used Simics to increase developer efficiency and better manage their risk by decoupling dependencies between software activities and hardware activities. The end result, simply put, is that our customers can now produce better software faster.

Through the context of the perfect project, we illustrated how allowing access to target systems, providing tools for collaboration, and automation can significantly improve all phases of your product lifecycle. We hope this paper has given you some ideas about how you can make your projects more perfect.

COLLABORATION AND COMMUNICATION

What does it mean when everyone uses the same set of tools and the same virtual replica of the hardware? It means everyone now "sees" the same thing, and that means they can communicate much more efficiently. (Are you familiar with the "well, it works for me" situation?) With "checkpoints," teams and individuals communicate and collaborate by sharing setups of the entire system and its state, all in software. They simply send an email to colleagues with the checkpoint attached and ask them to start execution from that specific point. Everything is included in the checkpoint, so the system can continue execution as if nothing had ever stopped. One individual can see and run exactly the same thing as his colleague. And it doesn't matter where in the world they are located.

WNDRVR