An Interview with SunSoft Chief Scientist Rob Gingell

One of the most popular presentations at the recent Solaris Developer Conference was given by Rob Gingell -- Chief Scientist at SunSoft™. Since so many attendees were interested in Rob's perspective on the future of the Solaris operating environment, and his detailed description of the Solaris API, the Catalyst Flash interviewed Rob for the benefit of all developers.

Rob has been with Sun Microsystems since 1985, and with SunSoft since its inception in 1991. He came to SunSoft to work on the Solaris operating environment, and has been responsible for a number of the key technologies in SunOS™ and Solaris; in particular the development of the ABI and its enabling capability of dynamic linking. The following interview gives an excellent insight into the internals of the Solaris operating environment and Rob's vision for SunSoft products.

Why do you think that your talk was particularly well received at the Solaris Developer Conference?

I think it has a lot to do with the fact that SunSoft developers are making critical technology decisions and want to know more about why Solaris is a good match for them today and in the future.

With that in mind, let's begin by discussing the Solaris API — the basis of your talk at the conference.

OK. Actually, the Solaris API is a slight misnomer. There really isn't a Solaris API, but rather a collection of a very diverse set of elements, including a traditional operating system, a window system, and a basic application desktop, with all the associated utilities, library routines, and documentation.

It's unusual to find one customer that is going to use all these facilities at once. The audience for these facilities is as diverse as the facilities themselves, with some people intent on building and delivering hardware, and others developing device drivers for add-on devices. Then there are the applications providers who are trying to create solutions for customers. After all, no one buys Solaris just to run it, but rather as a platform for solutions to solve their business problems. There are also all the volume end users who drive the dynamics of the whole market for products. These are the people who are trying to use information increasingly as a competitive weapon.

Across this space, we have a large number of APIs that are at different points in their lifecycle. Developers ask: “What are these lifecycles, how can I identify them and what does each stage mean? How do they end up affecting me?”

Those are good questions. How would you respond to them?

Well, while there are a diverse set of facilities, they are being delivered in a well integrated package that comes as a comprehensive platform. This means that customers can use a single platform to run their business and use it to drive both experiments as well as the basic set of operations that they are using day-to-day to run their business. There is a common base for the software you build and the software you buy. You don’t have to think about buying and learning to use different platforms for each of those capabilities.

Could you spend a minute explaining the elements of the Solaris API? Not all developers are completely familiar with its components.

Sure, I break it down like this. The basic elements include interfaces for the hardware providers, including DDI, DDX, and KBI. These are the interfaces between SunSoft as a software provider and hardware developers.

With these, we are trying to build the basis for product interoperability rather than just source interoperability.

There are also interfaces for those developers that are building standards-based products such as those based on POSIX, XPG, and the ABI. Standards support the attempt to have the maximum possible reach in the industry. If developers are doing this, it’s because they are trying to have the widest possible range of seats at which to target their application.

For those doing information presentation, there are interfaces represented by toolkits, window managers, and GUIs. For those trying to get the most out of the hardware, there are interfaces for new technologies such as multithreading which allow you to take advantage of the new generation of products such as deskside and desktop multiprocessors. We are also providing technologies such as Solaris LIVE!™ and Solaris VISUAL™ to exploit our multimedia and graphics capabilities.

Finally we have technologies that allow you to integrate all of this to create a coherent solution for running on a single system or distributed among many. We think that our networking technologies such as ONC+™, ToolTalk® for interapplication communication and Project DOE, object-based technology for interface management and communications between distributed services, will allow you to build more effective, interoperable, heterogeneous, and secure computing environments.

If you had to classify the components of the Solaris API, how would you do it?

Well, if you look at Figure 1, you can see how I view it. APIs can be examined in the context of three key issues: their availability (what is the volume you can achieve using them), how stable they are, and perhaps most importantly in terms of what SunSoft is doing with Solaris 2, is a set of validation technologies that we believe hold the promise of dramatically changing a developer's cost structure.

With every API we introduce, we are going to drive them with our own volume across as many platforms as we can. We are going to try to take these into the standards organizations so that they have the opportunity to become multi-vendor standards to help developers get the most out of their investment in Solaris. For example, we recently announced a major agreement with several key industry partners at Uniforum to develop a common software environment for all UNIX operating system platforms. (See the Common Open Software Environment article.)

In today's downsizing market where computer vendors are being compressed out, volume may represent survival. Developers will want to lean on volume vendors as they do their development because they want to make the investment with someone who is going to survive long term, and volume is likely to be part of that.

On availability, at least as far as UNIX vendors are concerned, Solaris is the only product that can take developers into the far right portion of the figure, where the volume cycle becomes realizable. Certainly all of the standards that Solaris supports allow developers to extend beyond onto other platforms; however, there really is no other UNIX vendor that is even close to hitting the million unit run rate.

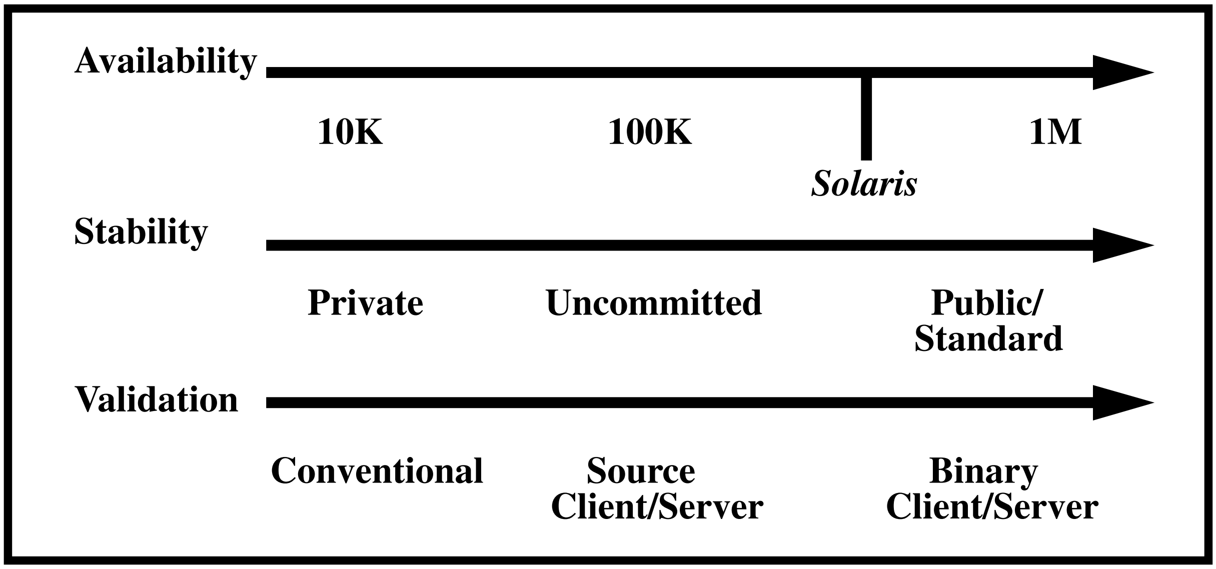

Figure 1. API Taxonomy Model

Certainly the business case is a critical concern for developers. What about stability?

Well, stability is an area that you will see being worked into our documentation. First, we have private interfaces that don't necessarily exist for developers to use, but rather are the basis for our internal development so that we can build Solaris. Of course the trouble with private interfaces is that they are private. They may move around. If you use a private interface, you expose yourself to a large amount of variance and risk. Developers will always want to avoid private interfaces and use only those interfaces that we tell them are intended for their use. The two classifications of interfaces we'd recommend to developers are uncommitted and public/standard.

Usually, "uncommitted" describes interfaces that are new. We provide these to help developers learn how to use them, and to help us learn how they ought to be supplied, so that they can be used more effectively. Occasionally we use the term uncommitted to refer to various public domain technologies, such as the popular "top" utility, that depends on uncommitted interfaces which we expect to change. Such products always bear the cost of having to track our implementation and thus they can be more expensive.

Finally, we use two names, public and standard. Standards are technologies defined by some public standards body that reflects the will of the industry and the attempt to save costs through standardization. With public, we refer to facilities offered in the Solaris operating environment, which while they are not standard, are treated as though they are. Hence, if developers are going to make an investment in our public technologies, they can feel confident that we are committed to them long term.

That all makes sense so far. What about your third criteria — validation technologies?

The third and final element in the API taxonomy model is validation, which tends to always be given short shrift within the industry. Everyone has rooms full of testers basically droning away beating on a product. This is where we think that the open systems marketplace can have a real advantage over the alternative marketplaces — by delivering to customers the open systems computing environment that they are expecting when the hear the words 'open systems.'

Most vendors currently provide conventional quality assurance, organized around vendors testing themselves independent of other products, simply trying to make sure that their products won't break. For example, before we didn't test what developers offer on top of the Solaris operating environment, which would allow us to tell whether they are using the interfaces correctly, or having them tell us that we have a bug of some kind.

This creates something known as the "bit 47 problem," where, over time, applications become dependent not on what the product is, but how it is built. This just doesn't make sense in a open system world. For example, imagine what it would be like if your car engine were dependent on how a specific brand of motor oil was made, as opposed to the generic properties of 10W40 weight oil.

Obviously you've never seen my car...

(laughing) OK, OK. It sounds like you have a unique problem. You can see my point though. This is the reason that many people look at the computer industry with such great frustration. They can see that the industry hasn't figured this out yet. Admittedly, software technology is a lot more complex than simple motor oil, but those are the properties that the customer is asking for from open systems. And these are the things that we have to find a way to deliver if we expect to have a pluralistic industry.

On the negative side, if we don't deliver these things, we won't have a pluralistic industry. The industry would just collapse to one or a small number of vendors and would be limited by the innovation that the one vendor could bring to the marketplace. History has shown that this model never works in the interest of the consumer. Hence, the very existence of a vibrant, evolving computer industry is dependent on our ability to forge '10W40-like' relationships between our products.

This is part one of a continuing interview with Rob Gingell. Watch for more in the next issue.