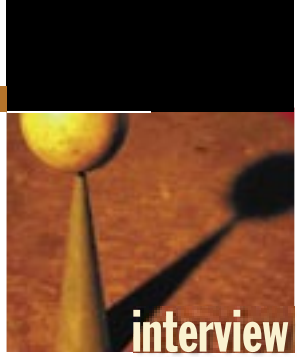

A Conversation with Werner Vogels

Photography by Craig Harrold

Learning from

THE AMAZON

TECHNOLOGY

PLATFORM

Many think of Amazon as "that hugely successful online bookstore." You would expect Amazon CTO Werner Vogels to embrace this distinction, but in fact it causes him some concern. "I think it's important to realize that first and foremost Amazon is a technology company," says Vogels. And he's right. Over the past years, Vogels has helped Amazon grow from an online retailer (albeit one of the largest, with more than 55 million active customer accounts) into a platform on which more than 1 million active retail partners worldwide do business. Behind Amazon's successful evolution from retailer to technology platform is its SOA (service-oriented architecture), which broke new technological ground and proved that SOAs can deliver on their promises.

Vogels came to Amazon from Cornell University, where he was working on high-availability systems and the management of scalable enterprise systems. He maintains that research spirit at Amazon, which regularly must solve problems never before encountered. "Maybe other companies call it research. We just call it development," he points out.

Interviewing Vogels is ACM Turing Award winner and Microsoft Technical Fellow Jim Gray.

JIM GRAY How has Amazon's technology evolved over the past five years?

WERNER VOGELS Growth is core to Amazon.com's business strategy, and that has had a significant impact on the way we use technology: growth through more categories, a larger selection, more services, more buying customers, more sellers, more merchants, more developers, increasing the different access methods, and expanding delivery mechanisms. The impact has been on many areas: larger data sets, faster update rates, more requests, more services, tighter SLAs (service-level agreements), more failures, more latency challenges, more service interdependencies, more developers, more documentation, more programs, more servers, more networks, more data centers. A large part of Amazon.com's technology evolution has been driven to enable this continuing growth, to be ultra-scalable while maintaining availability and performance.

Amazon.com started 10 years ago as a monolithic application, running on a Web server, talking to a database on the back end. This application, dubbed Obidos, evolved to hold all the business logic, all the display logic, and all the functionality that Amazon eventually became famous for: similarities, recommendations, Listmania, reviews, etc.

For years the scaling efforts at Amazon were focused on making the back-end databases scale to hold more items, more customers, more orders, and to support multiple international sites. This went on until 2001 when it became clear that the front-end application couldn't scale anymore.

JG Was that performance scalability, or was it manageability, or facilities?

WV The many things that you would like to see happening in a good software environment couldn't be done anymore; there were many complex pieces of software combined into a single system. It couldn't evolve anymore. The parts that needed to scale independently were tied into sharing resources with other unknown code paths. There was no isolation and, as a result, no clear ownership.

At the same time, there was continued difficulty in the back-end database scaling effort. Databases—and by that time we were using several databases—were a shared resource, which made it very hard to scale-out the overall business. So both the front-end and back-end processes were restricted in their evolution because they were shared by many different teams and processes.

We went through a period of serious introspection and concluded that a service-oriented architecture would give us the level of isolation that would allow us to build many software components rapidly and independently. By the way, this was way before *service-oriented* was a buzzword. For us service orientation means encapsulating the data with the business logic that operates on the data, with the only access through a published service interface. No direct database access is allowed from outside the service, and there's no data sharing among the services.

Over time, this grew into hundreds of services and a number of application servers that aggregate the information from the services. The application that renders the Amazon.com Web pages is one such application server, but so are the applications that serve the Web-services interface, the customer service application, the seller interface, and the many third-party Web sites that run on our platform.

If you hit the Amazon.com gateway page, the application calls more than 100 services to collect data and construct the page for you.

JG So my homepage on Amazon is probably 100 service requests?

WV Possibly. It depends a bit on what kind of page you visit—whether it is a product page, a checkout page, etc. It also depends on how effective caching is for the objects on that page, as well as some other factors.

The big architectural change that Amazon went through in the past five years was to move from a two-tier monolith to a fully-distributed, decentralized, services platform serving many different applications. A lot of innovation was necessary to make this happen, as we were one of the first to take this approach. Operating such a diverse set of services at this scale is not something that many people have done before, especially not with the kind of isolation that we wanted to achieve.

It has been a major learning experience, but we have now reached a point where it has become one of our main strategic advantages. We can now build very complex applications out of primitive services that are by themselves relatively simple. We can scale our operation independently, maintain unparalleled system availability, and introduce new services quickly without the need for massive reconfiguration.

JG Can you crystallize what you've learned from this? If you had to list three to five lessons learned, what would you say?

WV The first and foremost lesson is a meta-lesson: If applied, strict service orientation is an excellent technique to achieve isolation; you come to a level of ownership and control that was not seen before. A second lesson is probably that by prohibiting direct database access by clients, you can make scaling and reliability improvements to your service state without involving your clients. Other lessons are related to how you access services: If you want to be able to aggregate services easily, if you want to insert advanced infrastructure techniques such as decentralized request routing or distributed request tracking, you need a single unified service-access mechanism.

Another lesson we've learned is that it's not only the technology side that was improved by using services. The development and operational process has greatly benefited from it as well. The services model has been a key enabler in creating teams that can innovate quickly with a strong customer focus. Each service has a team associated with it, and that team is completely responsible for the service—from scoping out the functionality, to architecting it, to building it, and operating it.

There is another lesson here: Giving developers operational responsibilities has greatly enhanced the quality of the services, both from a customer and a technology point of view. The traditional model is that you take your software to the wall that separates development and operations, and throw it over and then forget about it. Not at Amazon. You build it, you run it. This brings developers into contact with the day-to-day operation of their

software. It also brings them into day-to-day contact with the customer. This customer feedback loop is essential for improving the quality of the service.

JG It's usually internal customers, though, right?

WV No, many services are directly customer-facing in our retail applications. Take a simple service such as sales rank; there is an attribute on most product pages that indicates what the sales popularity of that product is in its category. This is a separate service. It is a relatively simple service. The Listmania service, for example, produces specialized data that on almost every page is adapted to the specific product on that page and the history of the customer. This is a case where there is an important direct interaction between the developers of the service and the retail customer.

Amazon has very diverse groups of customers. The one very important group is, of course, the retail customers who shop on one of the many Amazon Web sites. Second, there are our retail partners. About a million small and larger businesses sell on the Amazon platform. For example, if you go to one of the book pages, you will find that item is also available new or used from some of our many partners. These can be very small independent bookshops or larger retail operators that want to sell on our platform.

Amazon.com is a very advanced e-commerce platform on which anyone can become a retail partner and instantly benefit from all the platform functionality that has made Amazon.com so successful. We make the Amazon.com technology and data available through the AWS (Amazon Web Services) e-commerce services. This is a free Web-services interface for developers, which they can use to build (and charge for) their own applications on top of Amazon. There are about 150,000 of these developers and we consider them important customers.

Other important platform customers are our enterprise retail partners who use Amazon services through a program that allows them to consume part or all of the services in the Amazon platform to build their own e-commerce Web sites or applications. Target, Bombay Company, and the NBA (National Basketball Association) are a few of our current customers, and this is a growing area for Amazon.

JG How many of the current buzzwords, such as SOA, WSDL, SOAP, WS-security, are relevant to you?

WV I would like to distinguish three categories of interfaces here. The first category is the services that make up the Amazon platform. There we use interface specifications such as WSDL, but we use optimized transport and marshalling technology to ensure efficient use of CPU and network resources.

The second category is the interface with our retail partners, which has strict descriptions for XML feed processing, service interfaces, etc., and where we leverage as many standard technologies as possible.

The third category is our public Amazon Web Services, which builds on the platform services and provides REST-like as well as SOAP interfaces. If we look at how developers use these interfaces, in general the REST version is used by small libraries in Perl or PHP as part of a LAMP stack, and the SOAP calls are mainly done by applications that have been built on Java or .NET platforms by consuming our WSDL files and generating proxy objects.

Do we see that customers who develop applications using AWS care about REST or SOAP? Absolutely not! A small group of REST evangelists continue to use the Amazon Web Services numbers to drive that distinction, but we find that developers really just want to build their applications using the easiest toolkit they can find. They are not interested in what goes on the wire or how request URLs get constructed; they just want to build their applications.

JG What about tools? Are you using Eclipse and Visual Studio as your dev environments?

WV There is quite a bit of development happening in Eclipse, but IntelliJ's IDEA is also popular for Java development. Some development happens in Visual Studio. Developers of our services can use any tools they see fit to build their services. Developers themselves know best which tools make them most productive and which tools are right for the job. If that means using C++, then so be it. Whatever tools are necessary, we provide them, and then get the hell out of the way of the developers so that they can do their jobs.

JG But if things are in XSD, then these tools work much better than if they don't have XSD. So the question I have is, have you done anything so that the tools dovetail with your standards?

WV I think part of the chaotic nature—the emerging nature—of Amazon's platform is that there are many tools available, and we try not to impose too many constraints on our engineers. We provide incentives for some things, such as integration with the monitoring system and other infrastructure tools. But for the rest, we allow teams to function as independently as possible. Developers are like artists; they produce their best work if they have the freedom to do so, but they need good tools. As a result of this principle, we have many support tools

that are of a self-help nature. The support environment around the service development should never get in the way of the development itself.

JG At this point, what are your biggest successes and biggest headaches?

WV Next to the things that we already talked about—the service-oriented architecture, the way we scale, the way we serve our customers—I think our biggest success has been that Amazon has become a platform that other businesses can benefit from. If you are one of the million retail partners on Amazon.com, if you're this very small bookshop somewhere in southern Florida, your products are immediately integrated into the Amazon.com recommendation system. You also instantly become part of our search system and customers can discover your products the same way they can discover Amazon.com products. All of the Amazon.com platform technologies become available to you as a seller on Amazon.com.

Making Amazon.com a general platform for e-commerce operations has been made possible through our advanced technology investments, and it has become a major success. Making Amazon.com available through a Web services interface to any developer in the world free of charge has also been a major success because it has driven so much innovation that we couldn't have thought of ourselves.

JG For example?

WV The mobile space is an example. This world is so much in flux, with respect to form factors, network speeds, and input methods, that it is difficult to build applications that give a good user experience across the board. We have made sure in our Web-services interfaces that developers can specify external XSLT (extensible stylesheet language transformation) stylesheets so that data can get post-processed at our servers and that what is returned to the consumer is in a format that is optimal for the device and application that requested it.

A cool application I saw recently in Japan runs on a camera phone. You can walk up to any product, make a photo of the barcode, send it off to the service, and it will come back to you with reviews, comparable products, and, of course, the price at Amazon. This was developed outside of Amazon.com using the Web-services interfaces. The small company that developed this is running a very successful service.

Developers have built a whole range of applications with AWS, from business applications to funky nonprofit tools. Our Web-service interfaces are very popular in the mixing and mash-up world. And there are software MP3 players that go to Amazon just to get the reviews and the image of the CD. We don't really mind. We see Amazon.com as part of the larger Internet ecosystem, and we want to stimulate innovation wherever possible.

JG What about headaches? What are the things that are driving you crazy?

WV One of the biggest challenges is developing at this large scale. How do you make sure that developers are productive in this large distributed service-oriented architecture? If you have this decentralized organization where everybody is developing things in parallel, how can you make sure that all the pieces work together as intended, now and in the future? An important part of that is, for example, testing. How do you test in an environment like Amazon? Do we build another Amazon.test somewhere, which has the same number of machines, the same number of data centers, the same number of customers, and the same data sets?

How do I test against this pair of jeans with that size, which just happens to be out of stock but will be back in stock two days from now, such that the customer e-mail interaction can be tested correctly? How do I test that with the new version of the offer-services that will roll out next week on the Japan Web site but with a browser that is not complete in displaying the Kanji characters?

These are very hard

questions, and we have no simple answers. Testing in a very large-scale distributed setting is a major challenge.

JG I'd like to back up a little and ask where your requirements come from. That is, how do you decide to do something?

WV There is, of course, long-range business-strategy planning—whether it is our future in digital media or the way we open up our platform for consumption by major partners—but there is also room for a lot of creative thinking by the larger Amazon community. It could be a developer coming up with a new business idea, it could come from the executives, or it could come from one of our customers. Our customers are very passionate about Amazon, and we give their input very high priority.

If an idea is deemed worthy of investigation, we exploit our service development approach to scope and prototype the idea quickly. With a new radical service, you try to go into prototype mode pretty quickly, and then you start iterating on that until you feel that you understand your business problem. The small-team concept means that you have a continuous feedback loop where you try to understand the impact for the customer.

That's in general how requirements are being refined, with the customer in the loop. It is also very important to try to determine at the outset what the success criteria should be, and how they can be measured.

This fast response to new ideas is enabled through the loosely coupled services model, both in technology and at the developer and operations level. From the outside, the services in our platform may appear chaotic, but chaotic in a good sense—in that we try not to impose a rigid structure on the different functional pieces, but we expect there to be order when looking at it from a different dimension. Thinking about this whole system as a big deterministic system would be unrealistic. Life is not deterministic, and a large-scale distributed system such as Amazon has many organic and emerging properties that can come to life only if you do not constrain it.

JG Let's postulate that somebody has come up with an idea and the team has gone off and built something. How does the go/no-go decision get made?

WV It may depend on the criteria for success that were defined up front. When the service is ready for beta testing, we will slowly introduce this to our customers, and then we measure relentlessly.

JG What kinds of things do you measure?

WV We have a very good understanding of how customers interact with the site as is. When we expose new features we measure how they change the customer's behavior. For example, does it take the customer fewer steps to find what he or she needs? This is hard because you are measuring human behavior; there are some things that customers are delighted about immediately and there are other things that they have to get used to.

A good example of a recent service is Statistically-Improbable Phrases, which can help customers find relevant items on many of our book details pages. We are

always looking for how we can give customers a richer way to find similar articles. Given that we already have the Search-Inside-the-Book feature, for which we have access to the text of many books, we are looking at how we can use these texts to expose different relationships between books. The Statistically-Improbable Phrases service looks for phrases in texts that appear to be unique for a book, and then builds indexes to find other books that may contain the same unique phrases. It turns out to be a mechanism that brings very remarkable collections together.

We measure whether or not a new feature is successful in terms of customer satisfaction: Do people find things more easily? If we can improve the convenience of shopping on Amazon, then we have booked a major success. If we can help them find things that they might not have thought of before, that is also excellent. Customers tend to vote with their wallets, so if there is a clear negative result, we know what to do with that service.

JG That's actually a very good example of a project that came from the inside and bubbled up.

WV Remember that most of our developers are in the loop with customers, so they have a rather good understanding about what our customers like, what they do not like, and what is still missing.

We have a lot of feedback coming out of customer service. Many Amazonians have to spend some time with customer service every two years, actually listening to customer service calls, answering customer service e-mails, really understanding the impact of the kinds of things they do as technologists. This is extremely useful, because they begin to understand that our user base is not necessarily the techno-literate engineer. Rather, you may get a call from a grandma in front of a computer in a library who says she wants to buy something for her grandson who is at college and who has a Wishlist on Amazon.

Customer service statistics are also an early indicator if we are doing something wrong, or what the real pain points are for our customers. Sometimes in meetings we use a "voice of the customer," which is a realistic story from a customer about some specific part of the Amazon experience. This helps managers and engineers to connect with the fact that we build many of these technologies for real people.

JG It used to be hard to integrate catalogs—and not just hard, but the results were catastrophically bad. It would be clear that things were from different data sets. Amazon has done a very good job of integrating lots of different organizations' catalogs. Why have you been successful?

WV Mainly, I think service orientation has helped us there. The stored data formats are decoupled from the format in which you communicate data items. If there is no need for sharing schemas of the actual storage layout, you can focus on making sure that the service interfaces can evolve in a way that allows you to handle variations of data formats. You could dictate a rigorous single format, but that would not be realistic if you are in Amazon's platform business. We have to make sure that the platform can be extended by our customers to meet their needs.

JG Are self-describing XML documents flowing around inside your service-oriented architecture?

WV We are on our way to using this in a number of cases, but we are not there yet. There are advantages, especially when you aggregate services into new services, or when you compose pipelines of services that operate on a message floating through the pipeline. But in many of our cases the contract between consumer and producer clearly defines the messages that will be exchanged, and we can optimize by not using self-describing features.

JG You've integrated with Sears, Nordstrom, Target, and so on. To what extent are you an extension of them and to what extent have you been involved in their problems? They have a bunch of legacy systems.

WV There are two categories of customers for whom this is important. First, there are the general retail partners, who are integrated with the Amazon.com experience. They provide us with item data and promotions, and we send them transactions. This can happen through a Web services interface or a specialized feed-processing interface.

But then there is the second category of large enterprise retail partners, such as Target, Bebe, or Sears Canada, to which we provide a large collection of platform services, out of which fully independent e-commerce Web sites can be built with complete personalized branding. Which services these sites consume differs from site to site—for example, some will use their own order processing pipeline and may or may not use Amazon's fulfillment services.

There is also a cross-over between these two categories; we can provide branded Web sites for the first category of retail partners whose data lives on the Amazon.com platform; and when customers check out their shopping carts, they enter the Amazon.com-driven order pipeline where they can use their Amazon.com identity to perform transactions. Diane von Furstenberg's dvf.com Web site is such an example.

These examples show some of the many ways applications are being built on top of the Amazon.com platform. There is Amazon.com and the different international

Amazon sites, there are the retail partners integrated with Amazon.com at the data level, and the enterprise partners integrated at the services level, and the many applications built on the public Web services interface, which all together means that a wide range of innovative applications are active on our platform.

JG You spend a fair amount of time recruiting. What's your recruiting strategy? How do you decide if somebody should work at Amazon?

WV The Amazon development environment requires engineers and architects to be very independent creative thinkers. We are building things that nobody else has done before, so you need to be able to think outside the box. You need to have a strong sense of ownership, because in the small teams in which you will work at Amazon, your colleagues will count on you to pull your weight—especially when it comes to operating the service that you have built. Can you take responsibility for making this the best it can be?

A very important point is whether candidates think the right way about customers and technology. Technology is useless if not used for the greater good of serving the customer. We are a strongly customer-oriented company, and we often use the "working from the customer backwards" approach. This means that in your thought process, you start with the customer and work your way backwards until you have found the simple and minimal technology that you need to satisfy the new customer requirement. It is important that engineers who come to work at Amazon understand that we do not build technology for technology's sake, but to support the customer.

JG You spent time at universities. What do you think about what they're doing now?

WV Different groups at Amazon interact with academia. Often a service needs to develop new revolutionary technology from scratch, and they will look at who in the research world worked on these topics before and who can help out.

As an example, at the infrastructure level we are building several systems that are a synthesis of some of the very exciting decentralized computing work that has rocked the operating systems and distributed systems world in the past few years. But we are finding that much of the academic technology is just not complete enough to be applied in real-life systems, as incomplete assumptions were often made. This may have been OK in an academic setting as it allowed for technology evaluation in isolation, but it just doesn't work in real-world engineering. The limitations of the physical world mean you have to work under certain constraints, regardless of how you would like it to be.

This requires us to do a lot of advanced development to fill in the gaps that these research technologies left behind. As a reward for this extra effort, we do end up with technologies that are extremely robust and are simple to deploy and manage.

JG Can people in academia help Amazon? What would you say about the current university situation?

WV I realize that it's hard in academia to do research at the scale of operation that Amazon requires. So we don't look to academia to solve those challenges for us. We're building data sets here at Amazon, however, to provide to academics so that we can get interactions going on some of the issues where they can contribute.

We have a number of internships and sabbatical positions where Ph.D. candidates, as well as professors, can spend some time in a very high-tech production environment. I really urge students to take at least one internship in a nonresearch environment, so that they can start to understand what it means to be effective inventors and how to develop technologies that can be used to build production systems. Doing your research at a research lab is certainly fascinating, but I find that the students who have come to Amazon for an internship find it extremely gratifying to be in the loop of building something real.

I have recently seen a few papers of students detailing experiences with building and operating distributed systems in a planet-lab environment. When analyzing the experiences in these papers, the main point appears to be that engineering distributed systems is an art that Ph.D. students in general do not yet possess. And I don't mean reasoning about hardcore complex technologies—students are very good at that—but building production-style distributed services requires a whole set of different skills that you will never encounter in a lab. These are skills your professor can't teach you because he or she never worked outside the lab either. If you really want to learn about building complex robust distributed services, an internship at Amazon will definitely give you that.

The same goes for the few professors who have spent some of their sabbaticals at Amazon. It has been eye-opening for them, and we welcome more of them. Q

LOVE IT, HATE IT? LET US KNOW

feedback@acmqueue.com or www.acmqueue.com/forums