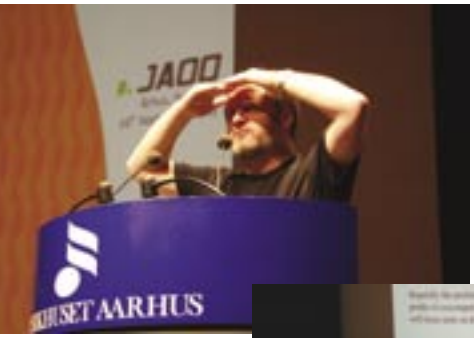
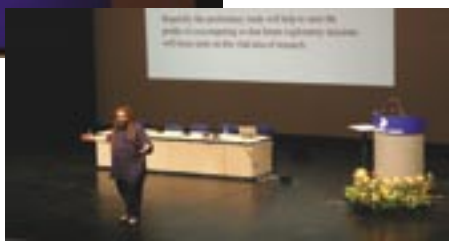
.JAOO TODAY

Tuesday, September 27





All photos from the first conference day...

Interview Raghu Kodali

1) What is your background?

I am product manager and SOA evangelist in the Oracle Application Server group. Apart from driving the requirements and feedback into the development plans, my job is to share the knowledge on new and emerging technologies with the developer community.

2) What is the nature of the problem that SOA is a solution to?

SOA is an architectural pattern that can be used and adopted to solve different types or problems or situations in IT organisations and enterprises. Interoperability, reusing or leveraging existing IT infrastructure, agility for changing business requirements are some of the typical problems seen in enterprises that SOA can offer a solution to.

3) It is often said that SOA principles can be applied inside and across organizations. Inside an organization, the IT infrastructure often consists of many heterogeneous legacy systems that are not service oriented. How can SOA principles be helpful in this context?

It's a very good question. A classic problem seen in the typical enterprise and reality out there in enterprises is that infrastructure is very hetrogeneous. Organizations have a wide variety of applications, middleware solutions, software packages, development and deployment platforms etc. This is where SOA would very nicely fit in. The basic idea of SOA is being "loosely coupled", and we need to have loosely coupled model in a hetrogeneous enviroment to cut down tremondous costs of developing and maintaining properitary integration between hetrogeneous components. By exposing business processes as services in a hetrogenous environment, we will be able to achieve the goal of connecting the hetrogeneous systems, without rewriting and reusing the exisiting infrastructure, which will also help us build and innovate newer systems that leverage existing infrastructure and adapt to ever changing business requirements in the enterprises.

4) Do you think Web Services is the right technology to implement a service-oriented architecture? Why or why not?

SOA is an architectural pattern while Web Services are Services that are implemented using a set of standards, and Web Services are one of the ways you can implement SOA. You can implement SOA on any number of technologies like messaging, passing around EDI docs, CORBA, etc., but lack of additional standards that go hand-in-hand with these technologies, and bolted on proprietary baggage that comes along with them (sometimes), more expensive infrastructure required (sometimes) to run these systems/services instead of running them on a bunch of commodity servers makes it difficult to harness the benefits of the architecture.

What Web Services brings onto the table is a set a standards that will provide a standard way to create a service contract that provides loose coupling between the definition and implementation, and several other standards such as WS-BPEL for service orchestration, UDDI for registry and SOAP for transportation to name a few.
General advantage of implementing SOA on Web Services is that you get platform neutral way of accessing the services and better interoperability as more and more Web Services specifications are supported by larger group of vendors.

5) Which top 3 sessions are you going to attend and why?

I definitely want to attend the following sessions:
(1) SOA - From Hype to Reality/ Nicolai Josuttis
(2) /Trends and the Future of Enterprise Java/ Floyd Marinescu
(3) /The Next Impedance Mismatch - Mapping Java Objects and Components to XML/ Dennis Leung

I definitely think that SOA is something that will help IT organisations and it is good to listen and learn from talks on how enterprises are deploying SOA. Floyd has done some excellent work on getting the news about the latest trends and happenings to the developer community. It will be great to hear his opinions on what he sees as the upcoming trends. EJB 3.0 is finally going to solve the impedance mismatch between database and java objects using POJO persistence in a standards fashion, but the mapping between Java objects and XML isn't being cleary defined yet, so this talk will be a good one to catch up on the issue and see the potential solutions out there.

Please see the interesting articel on Enterprise JavaBeans by Raghu Kodali on jaoo.dk

Interview Mike Keith

The annotation support included in Java SE 5 is being touted as something between a new and better way to develop applications and a full-scale programming revolution. With Java Standard Edition (Java SE) 5 now beginning to get a foothold amongst the conservative IT crowds, and the Java EE 5 platform planned for release soon it is worth taking a look at one factor that affects how we view annotations and XML.

I assume that the reader is familiar with annotations and program metadata, and how they are defined and declared. If not, I would recommend reading an introductory paper first, since this article will not introduce them or explain what they are.

It may or may not be obvious, but the primary difference between annotations and XML is where the metadata is located. The majority of their benefits and drawbacks result from metadata placement, and the consequences reach farther than one might initially suppose.

XML metadata is traditionally stored in flat files. Although it is possible to exploit file structure, such as using directory hierarchies, to offer more context this is seldom done. More often the XML is randomly lumped together as a wad of metadata with the subject of the metadata being explicitly provided in each case.

Consider a middleware layer that allows objects to be remoted (without implementation of any special “Remote” interface) and also allows certain methods to be selectively marked as remote. Furthermore, it allows specification of method parameters as being passed by reference when a call is local. To create XML metadata to uniquely identify such a method would require that it be fully specified. This would necessarily include the fully qualified class name, the method name and the set of ordered method parameter types. The extra contextual information is needed solely because the metadata and the artifact it is describing are spatially separate and must be associated. When attempting to read and interpret the metadata in its raw XML form the challenge is to differentiate the XML elements that have semantic meaning from those which are purely contextual.

Annotations are attached to the program artifacts they describe. This both gives relevance to the metadata when viewed and provides the multiple layers of context that must be explicitly conveyed in XML. The first and most apparent benefits of using annotations are therefore brevity and clarity. It is simpler to specify, and far less painful to understand.

Another advantage of coupling the metadata with the source code is practicality of the process. For example, if the application wants to change the name of its remote method, or change a parameter that is passed in then the developer must remember to keep the XML file (which may be stored in an entirely different location from the class) current. The XML file has a dependency on the code in that it refers to a method. This means that if the code changes there is a possibility that the XML may have to as well. The maintenance incentive for using annotations, then, is quite legitimate.

Similarly, XML that is not connected to the code is not an integrated part of the same version-controlled element as the code is. Changing one element and creating a new version of it does not intrinsically imply creating a new version of the other, although it is possible to configure some version control systems to do such a thing. Although there are cases in which changing one does not require changing the other, even a dependency in one direction (the metadata on the code) points to the more appropriate coupling of the two.

Annotations are built into the Java compiler and VM, so type checking at the Java class level comes for free. While it is true that XML schemas provide some degrees of validation and constraint checking they have a more manual flavor and are not nearly as sophisticated as the Java compiler. Being integrated with the language permits the pre-existing language infrastructure, such as classloaders and the reflection API, to be used to load and access the metadata.

Metadata – It’s All About Location

So why aren’t annotations used for all metadata if they are so wonderful? Well, they are clearly not the silver metadata bullet and do have their own share of problems that renders them less suitable for some applications or uses.

Take the example of a tool for adding metadata to existing classes. At the first step of the process, we immediately hit the first and most obvious problem. What if only the class files are present but the source code is not? Annotations may only be read at runtime, not added. Annotating the classes is not an option, and the tool is forced to use XML or some other mechanism that is external to the class.

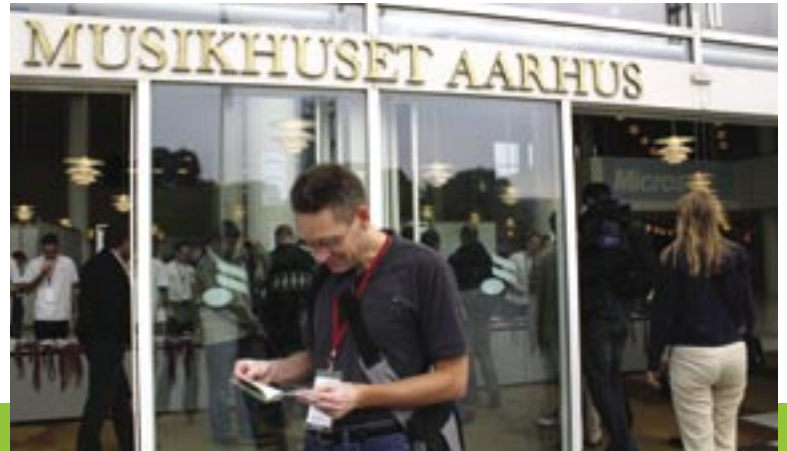
The next step is to actually add the annotations. This becomes a fairly intensive process for the tool, because the source needs to be parsed and the new annotations added at the correct position. Whereas using XML was a simple matter of having to specify the context and then write out the XML we now have to find the source code for the element, parse it, add the correct syntactical annotation pieces, and then rewrite it all back out. Interestingly, we end up with the inverse of the version control problem that XML had. Now we are forced to re-version the source element whenever we change the metadata instead of having the option not to.

Once the annotations have been added, the classes need to be recompiled. To achieve this the definitions for the annotations inserted into the source code need to be on the class path.

Although annotations, like an XML file, are quite happy to exist in runtime environments in which they get completely ignored, and the VM is more forgiving at class load time of elements that have annotations for which the definitions are not on the class path, this is currently problematic in current versions of the JVM.

The very notion of what an annotation is, additional information attached to an object, means there must be an object to which the information must be attached. One of the difficulties is that some metadata is more global in scope than any particular program element, meaning that there is not really a suitable object for annotating or situating the metadata. The only solution so far in this area is the package-info.java file, which allows annotations on a given Java package, but not globally. The deficiency lies within the Java language itself since it lacks an application or module artifact that could be annotated. This is being addressed in JSR 277 but won’t be ready until Java SE 7 (Dolphin).

Annotations are here to stay. They have made their debut on the Java stage and are now being integrated into the various Java specifications and standards. They are being heavily adopted and relied on for standardized metadata within the Java EE standards, such as EJB 3.0. Being able to use either annotations or XML, or even combining the two provides the best of both worlds and lets everybody do what works for them and for their application.



Interview Gil Tene

1) What is your background?

I've been working with various forms of virtual machines since the late 80's, and tackling scalability challenges for about the same amount of time. I built and shipped a number of products with several different companies over that period of time, spanning enterprise software, networking, security, and command and control systems.

I co-founded Azul Systems in 2001 with the intent of bringing dramatic new scale and predictability to datacenter applications. We set off to build an infrastructure class platform for delivering virtual machine compute capacity. I like to think of our systems as compute power stations, powering the grid of existing server running operating systems such as Solaris, Linux, HP-UX and AIX. In architecting and building these systems, we've tackled and solved some very fundamental problems around scaling Virtual Machine execution to 100's of CPUs and 100's of GB of memory, while delivering this capacity in a way that can be practically deployed. It is definitely the most creative and most fun period I can remember.

In the last four years, my team and I have created brand new garbage collection techniques, new multithreaded synchronization and execution techniques, a means of transparently delivering virtual machine compute capacity into existing servers and programs, and an amazing hardware platform that supports all these new features.

2) How would you define scalability, and in which ways can it be measured?

To me, scalability measures the ability of an application to deal with increasing workload while maintaining response or completion times at acceptable levels. Clearly, as more work is required, more resources are required. However, many applications reach a scalability limit where they are unable to effectively utilize additional resources even when they are available. In transactional and interactive applications, the most revealing external measurement is that of response time under varying rates of completed transactions. Another useful measurement of scalability is the efficiency with which an application consumes available resources. Tracking achieved throughout against available resources for an unbound workload will often demonstrate an application's inherent scalability limitations.

3) In which ways can virtual machines affect scalability?

Virtual machines can optimize for scalability and match the application to the underlying execution platform's capabilities. These optimizations can offer enhanced scalability to existing programs, often with little or no modification to code or configuration. Concurrent execution of synchronized blocks is only one example of how a virtual machine can significantly enhance scalability. Advances in garbage collection techniques in modern virtual machines allow applications to scale their memory footprint and throughput without compromising response times.

Enhancements in virtual machines and their use of 64-bit, SMP hardware are allowing us to move from the historic sweet spot of about 2 CPUs and 1 GB of memory per virtual machine instance, to tens and hundreds of CPUs, and tens of gigabytes of heap memory. It is now practical to deploy a 50-cpu, 20GB application instance that consistently responds to all requests in tens or hundreds of milliseconds. What we did at Azul is leverage a virtual machine as a way to ubiquitously provide such capacity to virtually any server in the datacenter, and to existing and future Java powered application or application servers. This capacity can be deployed as a shared asset, much like storage and network resources are. It can be accessed by applications and developers that would previously have been limited to the compute capacity they could afford in the form dedicated assets. I believe that ubiquitous access to well behaving, massively scalable compute capacity is going to open new doors to application developers, much like similar access to scalable storage and networking infrastructures did in the past.

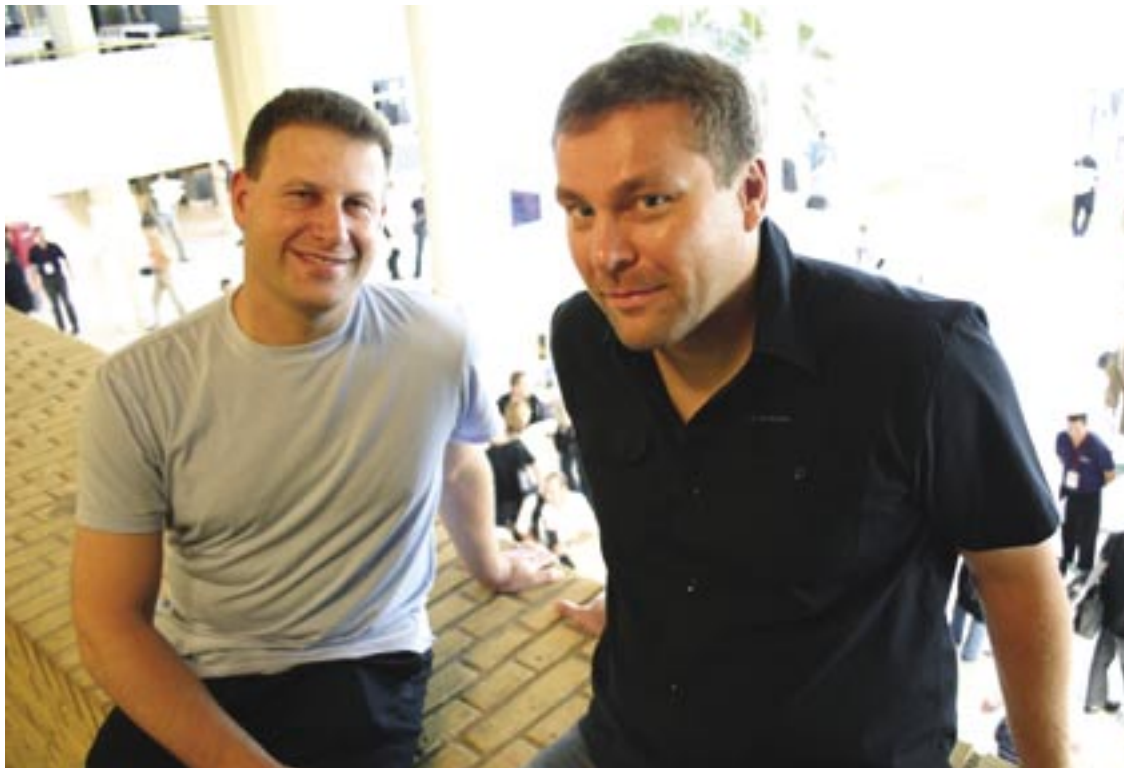
4) Building scalable applications can be hard. What are your favorite books on the subject?

Scalability is a very wide topic and spans many disciplines. For Java developers, I highly recommend a book I've been reviewing, titled "Java Concurrency in Practice" by Brian Goetz and Tim Peierls (with David Holmes, Josh Bloch, Joe Bowbeer, and Doug Lea). It will be available January 2006 from Addison Wesley, and I believe you can even preorder it now on Amazon.

5) Which top 3 sessions are you going to attend and why?

This is my first year at JAOO. I plan to sample the different tracks, and get people's advice on interesting talks to attend. On Wednesday I expect to be a bit busy, as my colleague Ivan Posva and I are presenting in three different sessions.

See article by Gil Tene on jaoo.dk about scalable computing and join Gil Tene and Ivan Posva for talks in the Java 5.0 and Scalable Computing tracks Wednesday.



Interview Frank Cohen

0) What is your background?

I've been writing software since the 1970's when I was a kid. I am one of Atari's first game developers, brought the Norton Utilities for Macintosh to market (along with Stacker and SoftWindows.) I was principal architect of the Sun Community Server and founded 3 start-up companies. Today I'm the "go to" guy for enterprises needing to understand and solve scalability and performance problems in their information systems.

1) Where do you see SOA being adopted in the real world today?

Recent mandates in the supply chain space in the US by Wal-Mart - the giant retailer - and in the defense industries by the US Department of Defense requiring information systems to use, store, and exchange data in XML format have software architects and developers asking the question: What is the role of data in an information system?

The resulting efforts deliver real world services using XML as the messaging medium and XML Schema as means to agree on the semantic knowledge of the message. I tested SOA implementations for General Motors where auto dealers order parts from the manufacturing plants using ebXML and UBL messaging, I see SOAP being used by an insurance underwriter as a general API to its insurance selling customers. I see RSS used in news feeds and podcasts.

In my mind, Web Services are all about XML messaging in a SOAP, REST, AJAX, or RSS environment. SOA is Web Services with a governance model. SOA answers the question: "Who will answer the phone call when the service breaks?"

2) What kinds of scalability and performance problems are enterprises encountering today in their Web Service deployments?

Many enterprises are failing to transition their SOA services from pilot to production. Enterprises are looking for rapid integration, flexible data management, interoperability and lower cost of ownership. Unfortunately, existing commercial and open source Java solutions do not perform well enough to become viable platforms for SOA development. Over the past three years I have measured scalability and performance of most of the Java application servers and found results in the 1.5 to 2 transactions-per-second performance level. That's just not good enough to get into production.

3) What do you recommend to solve scalability and performance problems and why?

I'm here at JAOO to introduce a new articture: FastSOA. The FastSOA architecture runs beside or in front of existing Web-base infrastructures. A service consumer (the client) makes a SOAP over HTTP request to a SOAP binding, which passes the XML data of the SOAP call to the XQuery engine. An XQuery processes the native XML request and may make queries to other services and data sources via JDBC, SOAP, and JMS protocols. The XML response document is passed to the SOAP binding and passed to the consumer.

Additionally, if the same document is requested multiple times, then the XQuery stores the response in a native XML in the mid-tier with time-to-live parameters. This delivers SOA acceleration through caching for even faster SOA performance. The goal of FastSOA is to deliver an order of magnitude faster performance and scalability.

4) How does XQuery and native XML database technology help a Java developer deliver well performing and scalable SOA?

Java tools normally build a proxy as a binding between a bean and the HTTP-based service interface. The problem is that the binding is normally very inefficient. I've seen some bindings create 15,000 or more objects to handle a SOAP request of less than 25,000 bytes of message payload. These transformations kill performance!
It seems natural to me to expose a SOAP service that when called executes an XQuery to handle the SOAP request. In an XQuery environment there is no transformation into objects. Plus, I've seen XQuery implementations that deliver extensions to the specification that enable the XQuery to make a call to a Java method. (Raining Data TigerLogic does this.) So you've got the best of both worlds: the XQuery handles the SOAP request and if needed the XQuery calls a Java object for additional processing.

Native XML database technology helps a Java developer to achieve faster performance and better scalability in persisting XML data. Using a relational database requires the XML to be transformed into relational tables. That transformation kills performance too. Adding XQuery to a native XML database enables mid-tier caching and service accelleration and data mitigation and aggregation services.

5) Which SOA related books can you recommend?

I really like David Chappell's book The Enterprise Service Bus. It is well written and covers most of the basics.

Of course I am shameless at plugging my new book FastSOA (aka Real World XQuery) that Morgan Kaufmann will publish next year. Chapters are available for free download at http://www.xquerynow.com/thebook

6) Which top 3 sessions are you going to attend and why?

I plan to attend...

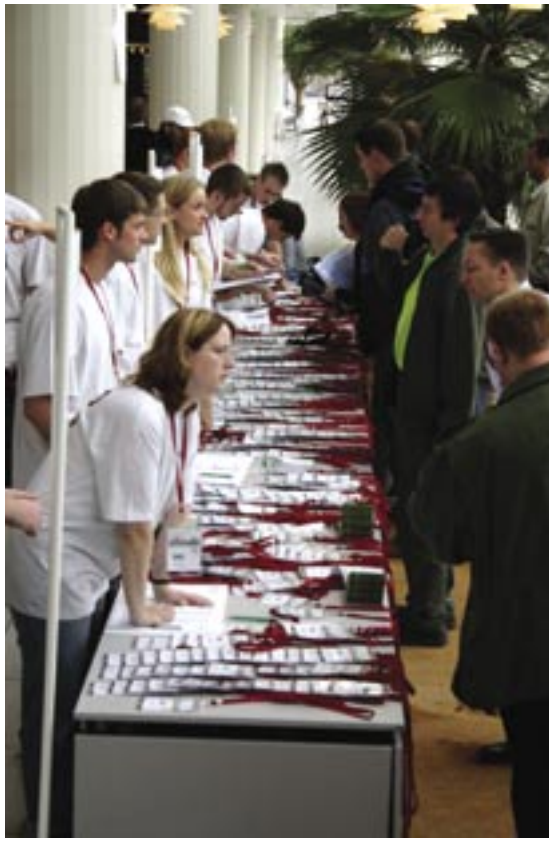
Ivan Posva's presentation on Presentation: "Java Technology Performance Myths Exposed", Track: Java 5.0, because I want to learn what hundreds of Java CPUs can do for performance of an application and service.

Jim Hugunin's presentation on "IronPython: Python on the .NET Framework", Track: Scripting And Dynamics, because Jim created Jython - the scripting language I embed in my TestMaker open-source test tool.

Tim Bayen's presentation on "Presentation: "Workflow, BPM, orchestration and Java", Track: J2EE, because this is the year of BPEL. Well, isn't it?!

7) What surprises would you like to find at JAOO?

I would love to see Azul Systems announce a 200 CPU laptop. (And I'd also like to know where the Jet Pack that Boeing promised me at the 1965 World's Fair is!)



Interview Ivar Jacobson

Today everyone wants to be agile. It is pathetic to see how basically every speaker has added the word agile to the title of their talks. I haven't done it. I haven't done it, for two fundamental reasons.
1. All my work over the years has, as I soon will explain, strived to make software development more agile.
2. Agile is not enough, we need more . I want agile+++, the meaning of the pluses I will explain in a minute.

Thus we all agree that we need agile software processes. We all agree on many agile principles such as iterative development, continuous integration & testing, use only what you need, etc. We agree with the four statements in the agile manifesto adopted by the Agile Alliance.

· Individuals and interactions over process. Of course, people are more important than the steps laid out and described in a book about process, since books don't produce software. However, people need to take advantage of explicitly expressed knowledge in order for a software project to succeed consistently.
· Working software over comprehensive documentation. Of course, working code is the only thing that's guaranteed to make your customer happy, no matter how rigorously your design and architecture might be expressed (for instance in UML). However, the code needs to be understood and maintained after your initial development team has moved on. This is hard with code-centric models; proper use of the UML makes a big difference in this context.
· Customer collaboration over contract negotiation. Of course, software requirements are very difficult to specify at the beginning of a project; they evolve as the software goes through iterations. However, the requirements need to be there for the customer's future reference (the customer speaks English, not Java code).
· Responding to change over following a plan. Of course, detailed planning at the outset of a project is likely to cheat the customer down the line. However, a rough overall plan and small plans for the next small steps help keep the project at its target area.

However, we have different ways to get there. The so called "agile methods" primarily rely on tacit knowledge. Tacit means implicit knowledge (achieved ad hoc and undocumented). The Unified Process relies primarily on explicit, structured knowledge. This is a big difference that has not come through in the debate.

A process or a method being agile to many means that its definition (description, book, web site, etc.) is light, i.e., it is sketchy. Thus when working according to it in a project, you have to use tacit knowledge and therefore the project is agile. This may be true in many cases, but in larger organizations you usually have a lot of people with different tacit knowledge. This creates a lot of difficulties working together. People need guidelines in the form of explicit knowledge to work consistently and create good software. Thus, a very important insight is that:
A light process may make a project heavy.

We also know that too much explicit knowledge such as in the Unified Process (UP), can make projects heavy, if they have to select what to learn, learn it, apply it and update it with new explicit knowledge as they learn more.

Thus, on the one hand it is hard to see how methods based on tacit knowledge (read agile methods) can scale into the future.
· We can't teach people much, since we have little common knowledge to teach.
· We can't grow our knowledge base, since there is no base to grow.
· We can't build tools based on our knowledge, since we don't know what knowledge to support.
· Etc.

Beyond Agile: Smart

On the other hand methods based on explicit knowledge (read Unified Process based methods) and delivered in some form of book (configurable and extendable or not) are overwhelming and can't scale either but for a different reason. Here the heaviness and the cost are delimiting factors. Only a small part of the software community will adopt them.

However, if we in some "magical" or say smart way dramatically can reduce the work to select, learn, apply and update the knowledge in UP, the situation will be different. If we can deliver the knowledge you need, and only that knowledge, and exactly when you need it and not before, then the size of the process doesn't matter. Whether the process "book" is 100 pages, 1,000 pages or for that matter 100,000 pages will be irrelevant for ease of use. Thus, the more the better! You will only get just what you need and when you need it. And the bigger book, the more real on-line mentoring you will get.

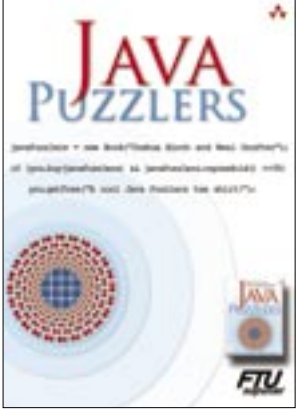
To make a process smart we need to codify the explicitly captured knowledge in the form of intelligent agents. Every developer has an online agent, or as we say, a virtual mentor. The virtual mentor is able to select what you need know, teach you exactly that, help you apply what you learnt and learn itself from your experience. We have proven that this technology really works (see for example Jaczone's WayPointer). Much more will happen in the years to come, but already today, as experienced by Tata Consultancy Services, substantial increases in productivity (more than 20%), quality, user experience, etc. can be made.

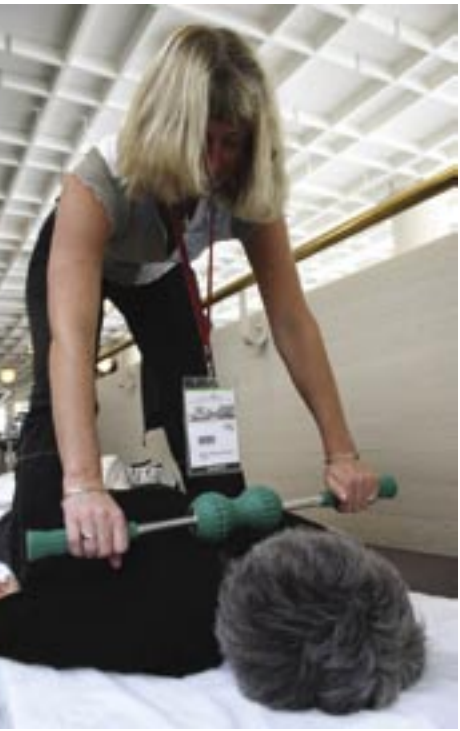
XP talks about pair programming. With these virtual mentors we talk about:
– Virtual Pair Programmers
– Virtual Pair Analysts
– Virtual Pair Designer
– Virtual Pair Tester
– Virtual Pair Project Managers
– Etc.

A smart process is agile (as defined by the agile manifesto) but it is also more. I have formulated four manifesto-principles to define a smart process and I call them Manifesto for Smart Software Development:
· Explicit knowledge over tacit knowledge. We should not need to spend time getting people to reinvent well-known stuff, nor should we waste time explaining it. Knowledge should be made explicit, as well as easily accessible and learned.
· Active process over passive process. Development teams no longer need to see process as something static that needs to be learned. Instead, the process works together with the practitioner actively as peers.
· Team capability over dependency on individuals. Instead of letting knowledge reside only in the heads of some key individuals, knowledge should be shared by the team. The team should share the work load as equally as possible.
· Self-organizing teams over extreme (rigid or lax) organization structure. Today, teams tend to err on two extremes - being too rigid in the way they work, or being too lax to the point of loosing control. The process should be flexible without loosing control.

I can't see that methods or processes that primarily rely on tacit knowledge ever can compete with smart processes. Together I hope we can make the software world smarter and not just agile.

Also at JAOO conference 2005





Book Signing

Tuesday 12:30
Ivar Jacobson, Ted Neward, Gilad Bracha, Frank Cohen and more from Pearson Education.

```
javaPuzzlers = new Book ("Joshua Bloch & Neal Gafter");
if (you.buy(javaPuzzlers) && javaPuzzlers.copiesSold () <=35)
    you.getFree("A cool Java Puzzlers tee shirt!");
```

Tuesday 15:30
Come and have a glass of wine and get some books signed by authors from John Wiley.

Wellness

How can boost more energy into your body?

Drink water with energy.
Breathe air without acid.

Programme your brain for better sleep.
Use the latest technology to reduce your sleep.
Try bioaxial rotation to recharge your batteries and you generate better code.

How can you improve your balance?

Come and see us and get a free energy boost and Pi-water - we are right next to the information booth.

Organizer of JAOO

EOS
EASTFORK OBJECT SPACE