JAOO TODAY

Wednesday, September 28

Organizer of JAOO

EOS
EASTFORK OBJECT SPACE

Brian Barry Commentary

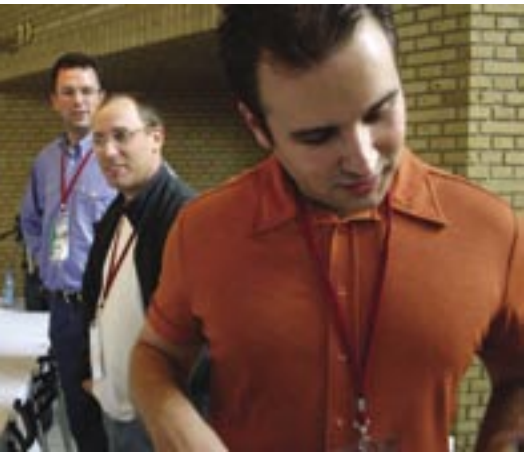
The morning began with Ivar Jacobson's plenary talk "Beyond Agile, Smart". Going in I was very curious as to exactly how this was going to go, and where Ivar would take with the subject matter. As it turned out, his main point was that most software engineering methods shared common principles, and when reduced to essentials depend on a combination of experience and knowledge. That knowledge can be tacit (by which he meant implicit, non-formalized) or explicit (formal, structured). He went on to associate Agile methods with tacit knowledge and the (Rational) Unified Process with explicit knowledge. Obviously in both cases practical experience is needed to be successful, so it is the type of knowledge that each method draws on which differentiates them. From this base he launched on a more controversial point, which was to assert that RUP is really better than Agile (because explicit knowledge is superior to tacit knowledge), but suffers from a delivery problem: it requires to too much effort to manage the RUP knowledge base (manage in this case meaning to find , learn, apply, and control knowledge). Ivar's solution to reducing that cost is to use intelligent agents to actively mitigate access to software engineering knowledge. The presentation was peppered with anecdotes from his career as a developer, project manager and methodologist, and altogether quite entertaining.

I believe there is a fundamental split in the software engineering (SE) community between those who believe that SE knowledge (i.e. how to build and maintain good software systems) can be codified, organized and transferred (Ivar's camp), and those who feel that it can't (my personal bias). The latter group is inclined to feel that the best (only?) way to become a good SE is by apprenticing with a master, who can guide you but can't provide an explicit rendering of his personal knowledge base. In other words, SE has more in common with medicine than with say, accounting. It's an interesting debate that will not be resolved any time soon.

I spent the remainder of the day in the Sixth Generation Language track (since I was speaking later in that program). Dave Thomas led off with a condensed history of programming languages, briefly characterizing each language generation and mentioning a number of outstanding exemplars. He asserts that the sixth generation languages are those that will support MDD (Model Driven Development), which includes DSLs (Domain Specific Languages), MDA (Model Driven Architecture) and DOP (Domain Oriented Programming). All of these will be needed to build applications in the coming era of Real-Time Business, where computing resources (CPU cycles, memory, bandwidth) are free, there is an overwhelming amount of data to process, and answers must be available in (near) real-time. This talk set the stage for the remainder of the session.

Dave was followed by Richard Soley, the head of OMG, who provided a general introduction to MDA. Richard observed that 90% of software costs derive from maintenance and integration of existing software, not from new development. OMG's mission is developing standards to help control and reduce these costs, and MDA is a primary vehicle for pursuing this goal. He presented MDA as a logical next step in the progression of programming languages, an attempt to raise the level of abstraction, using a graphical language, UML, together with MOF-based transformations (the analog of compilation) between abstractions (models).

Richard was followed by Erik Meijer, who did a great job of convincing us that there were some really exciting things happening from a language perspective in the next release of Visual Basic, VB 9.0. Many of these new language features show heavy influence from work done in the functional language community (e.g. Haskell). Perhaps the most interesting is the effort to integrate query operations into VB, making XML a first class data type, which (they hope) will make XQuery obsolete. VB 9.0 will have a full SQL-like language. Other new features include anonymous types, nested functions (closures), better support for Arrays (initializers, anonymous arrays), local type inference and dynamic interfaces (impose structure on late bound objects).



Floyd Marinescu

Floyd Marinescu and Mike Keith gave talks targeted more specifically towards the Java community. Floyd covered the hot new trends that are getting attention from Java developers, such as AOP, dependency injection, annotations, JVM based scripting (Groovy, JPython), AJAX and Rich Client technology, EJB 3.0 and the rise of Eclipse as the de facto IDE standard for Java developers. Mike Keith (a former colleague of mine from OTI days), spoke on techniques for mapping Java objects to XML. Mike's background is on the TopLink object to relational mapping product, so this is an area he has been working on for some time. He explained JAXB, and then did a "compare and contrast" between the Object-Relational and Object-XML mapping problems. He feels that JAXB represents a good start but it does not solve all the problems.

The other talk in the session was my own concerning "Trends in Open Source". I went through as brief history of the open source movement, tried to characterize why open source projects succeed or fail, and then made a few prognostications about how open source may shape the software landscape over the new decade or so. In particular I think that while the free availability of open source infrastructure like Eclipse can reduce startup costs for new programming language projects, it will also create a difficult business climate that may not result in much funding for innovation in the language area.

Platinum Sponsor

Microsoft®

Sun microsystems

Gold Sponsors

ORACLE®

SAP®

Silver Sponsor

Trifork

Bronze Sponsors

InterSystems

Google

RainingData®

Throughout the JAOO conference FTU Boghandel in corporation with Pearson Education and John Wiley offers all particiant a special 15% discount on all books.

Ted Neward

What is your background?

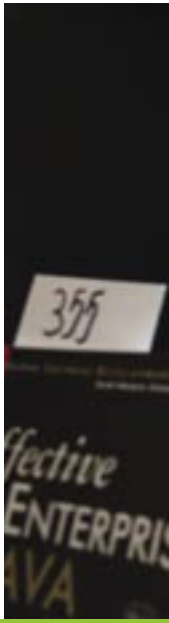
I'm an author, consultant and mentor, pecializing in enterprise development for 10 years, with experience (and interest) in C++, Java, .NET and XML services.

You have written several books about Java and .NET, and in general you know a lot about both worlds. Can the Java world learn anything from the .NET world and vice versa?

Absolutely. There's a lot of lessons being learned in both directions, and that's a healthy part of a competitive market--.NET is learning the Zen of managed code from the 10 years of experience that Java has, and Java is learning a lot of innovative approaches to those same problems from .NET. For example, .NET brought the idea of custom metadata attributes into the language, which Java "leveraged" as JSR 175, whereas Java's experiences with object/relational mapping tools and frameworks taught Microsoft a great deal about how they wanted to approach this problem, culminating in the recently-announced Project LINQ.

Java has been around for about 10 years and in that amount of time, a strong developer community has emerged. How strong would you say that the .NET community is compared to the Java community?

This is actually a pretty difficult question to answer, more so than many people might think. Obviously, if you measure the strength of the community by the number of open-source projects within that community, .NET has a ways to go to catch up to the Java space. If you measure the strength of the community by the number of conferences and their size, then it's pretty clear that the .NET community is actually stronger than that of the Java space. If, on the other hand, you try to measure it by the number of developers using the language(s) on a daily basis, you then have to wade through the (obviously biased) numbers put forth by both Microsoft and Sun, both about their community as well as each others'.



Robert C. Martin (Uncle Bob)

Why do you call yourself Uncle Bob?

It was a nickname given to me by an associate 17 years ago. I hated it at first; but once I became a consultant I missed it. So I added it to my signature line. Apparently it stuck.

Are agile methods best suited for small, co-located teams or can they be used by large, globally distributed teams?

Agile methods can be adapted to virtually any size team. We are working with companies of all sizes. For very large organization, some extra practices are necessary, but the essense is unchanged. We work in short cycles, with lots of feedback, and communication.

What about outsourced and offshore development teams, where organizational boundaries, language and cultural differences, and other communication challenges are likely to be encountered?

Physical separation makes it more difficult to do Agile. This does not mean it's impossible, or even inadvisable; it just means it's a bit harder. It requires more coordination, and an extra effort to communicate well. It is essential that the separated folks share a common mental framework, even if they don't share a common physical space.

There is another factor to consider. Agile methods reduce the cost, and increase the efficiency of local software development. This improvement compromises the economic advantage of outsourcing. Indeed, we may very well see a shift in policy towards building software at home as Agile Methods become more and more prevalent.

How does agile development fit into an organization having its own test/QA team?

QA and Test are the groups that experience the most profound changes in an Agile Transition. QA moves from a back-end verification role to a front-end specification role. Moreover, QA and Test are matrixed into the the teams as opposed to operating separately. There is no "throw-it-over-the-wall" concept in Agile Development.

Ted Neward

Mary Poppendieck A History of Lean

In 1945, Toyoda Kiichiro challenged his struggling automobile company: "Catch up with America in three years; otherwise the automobile industry of Japan will not survive." The man who led this effort was Taiichi Ohno, who is widely known as the Father of the Toyota Production System. Ohno knew he had to increase productivity by an order of magnitude, so he adapted the assembly line concept pioneered at Ford to the small market reality that was post war Japan.

Ohno discovered what Michael Dell discovered decades later as struggling college student in Austin, Texas: for a company with limited means and unpredictable customers, the only option is to "make a little, sell a little." So Ohno figured how to eliminate the large batches of parts that fed the typical American assembly line, while Dell figured out how to eliminate the middleman in the distribution chain. In both cases, the approach was as unorthodox as it was successful: Both Toyota and Dell have cost structures that are 25– 50% lower than their peers.

In the book "Toyota Production System," Ohno focuses on two keys to Lean success. The first key is well known: Just-in-Time inventory flow. The second key is not as well known: Stop-the-Line culture. Driving down inventory brings problems to the surface that must be acknowledged and dealt with immediately. Unfortunately, well-meaning people often "work-around" these problems or patch up their symptoms rather than eliminate the cause. Ohno suggests that a culture which tolerates "work-arounds" is at odds with the Lean approach of driving down inventory to expose problems so that they can be investigated and solved.

You can't have a Just-in-Time flow without a Stop-the-Line culture. You cannot reliably and repeatedly go fast if you encourage people to work around ambiguities, accumulate defects to be dealt with later, or wait until verification to uncover problems. Establishing a Stop-the-Line culture in a manufacturing plant was often accompanied by the slogan "Do it Right the First Time." In practice, this meant having tests in-line at every point of manufacturing so as to discover any incipient problems the moment they occurred, and then stopping to fix the problem immediately, so as not to make a lot of bad product.

Unfortunately, the "Do it Right the First Time" slogan made its way into development, where it was misinterpreted as a criticism of the natural learning cycles found in a good development process. So instead of driving down our inventory of work-in-process in order to uncover and attack problems, we build up large inventories of tentative requirements, untested code, and un-integrated modules. Unfortunately, a slogan which was coined to encourage people to build quality into a product has been used to encourage us to do quite the opposite.

Returning to Ohno's two principles, the Lean approach combines a Just-in-Time flow with a Stop-the-Line culture that surfaces problems and encourages workers to stop, investigate, experiment, and improve their product and their processes as a normal part of every

A Just-in-Time flow encourages us to move from customer need to deployed software as rapidly as possible, with work packages that are as small as possible. But at the same time, we must not tolerate the problems, defects, or ambiguities, which naturally arise as we drive down our inventory of partially done development work. In addition to Just-in-Time flow, we need to create a Stop-the-Line culture in software development.

The first indicator of a Stop-the-Line culture is a very low defect rate. If you routinely find defects after code should be working, you are testing too late. The second indicator of a Stop-the-Line culture is the absence of requirements churn. If you spend a lot of time changing requirements after they are written, you are writing them too soon! The ideal approach is to write "executable tests" in place of requirements in a Just-in-Time manner, maintain a single code base and integrate new code several times a day, run a test harness immediately, and stop (or at least stop checking in new code) if the tests don't pass.

Everything you already know about good planning, good architecture, good customer understanding, and good verification remains as valid in a Lean organization as anywhere else. But Lean had no use for plans that are mistaken for fact, architecture that substitutes detail for vision, requirements that protect from blame rather than foster collaboration, and verification that displaces in-line testing. Finally, Lean has no place for processes or specifications that discourage workers from quick experimentation and continuous change, because in the end, Ohno tells us, Lean is all about learning.

Jutta Eckstein

How did you get involved in the IT industry?

While completing the basic studies I had my first encounter with IT - I had to learn Turbo Pascal. I was so excited about programming, that I thought this is much more creative then industrial design could ever be. From then on I focused on software engineering.

What does managing an agile process feel like?

In short: Physically exhausting ;-)
The reason I gave that short answer is, that the management of an agile process is directly related to management by walking around. It is a lot about actively asking for feedback.
If you don't go out and look for yourself you will never know how the actual process is really suiting the people on the team. And furthermore you will have a hard time figuring out how the process has to change in order to fulfill the needs of the team.

How do large-scale enterprise applications fit in with agile processes?

Very well, and moreover: Using an agile process will be very beneficial for a large-scale enterprise application!
Agile software development will reduce all risks enormously, by ensuring feedback about the real and actual system all the time (and not only at the very end of the project).

Of course if you are wondering how you can make the shift to agile development with a (large?) team developing a large-scale enterprise application, well then I have to recommend to read my book ;-)

Christian Weyer

You are a recognized XML, Web Services and service-orientation expert within the Microsoft world. What is your background?

I am co-founder of thinktecture, a small company aiding software architects and developers in realizing projects with .NET and distributed applications technologies.
My career started on the Java platform a long time ago and turned over to the Microsoft world by hacking VB6, ASP and C++ COM. Over the years I evolved to a distributed applications, XML, Web Services and service-orientation expert.

I have worked for many years with Microsoft technologies like COM/DCOM, COM+, and last but not least: .NET. Since the very first days of .NET, when it still was called NGWS, I was working and writing with and about .NET and its related visions and technologies.

What's cool and not so cool about service-orientation?

Now we start to have the technological tools to make the basic vision of component orientation come true. Not so cool: The hype, the over-hype. Ask 20 people about SOA and you get 25 opinions.

There is a lot of hype about domain specific languages. Why are they interesting?

First, DSLs are nothing new. They have been around for a while. Second, I think Microsoft is creating a buzz (again). Almost everytime MS talks about DSLs they mean something graphical inside of Visual Studio. Obviously, this is not just it. The really interesting part
about DSLs nowadays is that they are accompanied by tools to build and enable them.
DSLs are not just for a technical domain.
They are also rich to provide an excellent communication means with the end user. Using DSLs to model the domain of the end user, his business world.

Ralph Johnson

You are one of the leading pattern experts, and an expert on software reuse and object-oriented design. How did you achieve this?

When I came to the University of Illinois, my goal was to learn how OO programming changed the way people programmer, and especially how it enabled reuse. I worked with anybody who wanted to build software in and object-oriented way. I quickly learned that using an object-oriented programming language didn't necessarily result in reusable software. Early on, I learned about frameworks, and started working with them and designing new ones.

The way to learn about design is to design, and to study others designs, and to get feedback on your own designs. It is important to think about what went right and what went wrong. People in industry often have too many deadlines to take time to think. People in universities often don't get enough practice to have something to think about. I was in the good position of both being

The way to learn about design is to design, and to study others designs, and to get feedback on your own designs. It is important to think about what went right and what went wrong. People in industry often have too many deadlines to take time to think. People in universities often don't get enough practice to have something to think about. I was in the good position of both being able to practice and being able to think about it.

Which books about patterns are your favorites?

Eric Evans book on "Domain Driven Design" and "Patterns for Time-Triggered Embedded Systems" by Michael Pont.

The Windows Workflow Foundation

What is Windows Workflow Foundation?
Microsoft Windows Workflow Foundation (WWF), which we announced at PDC 2005 two weeks ago, adds workflow to the Windows platform. It will ship in WinFX, which is free by the way. WinFX also includes Windows Presentation Foundation, formerly Avalon, and Windows Communication Foundation, formerly Indigo.

WWF has the potential to enable every single application that you have ever written on Windows to have workflow capabilities - a bold statement indeed, but by adding this capability to the Windows platform we enable all the developers out there to add workflow directly into their applications whether those applications are console applications or web services applications or Windows form applications or future generation applications. In fact thinking about it another way, WWF adds a new pattern to mainstream application development - I call this the "externalized workflow-first" design pattern.

Think about it, when you build UI you just reach for WinForms and/or the Windows Presentation Foundation; for data management you grab ADO.NET; and for distributed connectivity and messaging you use Enterprise Services, WSE, and/or Windows Communication Foundation. But for business services and business objects what do you do? We're starting to see the emergence of light, flexible modelling approaches based on DSLs to address this problem. They can give you a great start especially when used in conjunction with frameworks. But we still also see a huge amount of code that is simply branching logic - "if", "else," "while loops" and so on, this is opaque and is effectively workflow hard-coded directly into applications. Now imagine we are able to externalize the "workflow" aspects of applications and make them explicit through models. Suddenly an essential part of applications will be freed of their shackles - they will become more transparent. Furthermore, by providing graphical construction tools, we can make the workflow modelling experience even better. Try to imagine how powerful it would be to tie workflow models together with domain specific models in a DSL. I find that thought really exciting!

providing graphical construction tools, we can make the workflow modelling experience even better. Try to imagine how powerful it would be to tie workflow models together with domain specific models in a DSL. I find that thought really exciting!

What is the development experience of using Windows Workflow Foundation?
Workflow is going to be new to a lot of developers so we've tried to make everything about building workflows as seamless as possible and easy to understand for .NET developers. We have an API set in System.Workflow to directly program workflows. But we also have a designer for developing workflow models integrated into Visual Studio 2005. This means you can build workflow models in Visual Studio and create workflow artifacts that become a regular part of your solution projects. You compile these as usual because they actually end up becoming VB.NET or C# code through model-based code generation.

The designer allows you to create the fundamental workflow component called activities. From these you can compose sequential workflows and event-driven or state transition based workflows. Behind the scenes, but not hidden from view, the designer is actually modifying the source code for the workflow which is comprised of statements which create the workflow as a model. The workflow ends up as a collection of activities with properties set via the designer. When run the generated source code creates your custom workflow type in memory and executes it on the workflow runtime. Since the workflow runtime has control of that type, it can manage both the lifetime of that workflow and the state that's associated with it. The lifetime of the workflow need not be short like a procedural piece of logic, it can be long-running since it may be waiting for events from other parts of your application.

You can actually use an XML representation, XAML in fact, to create a workflow. So, it's very consistent with other technologies we have on the .NET Framework or in WinFX. Our customers demand transparency so you can literally write code and then load it into the designer that will

→

consistent with other technologies we have on the .NET Framework or in WinFX. Our customers demand transparency so you can literally write code and then load it into the designer that will reverse engineer the code into the graphical representation of the workflow. This is good for hardcore developers who prefer code to graphical diagrams, but it also makes it a lot easier to learn this technology. Having this graphical-textual duality is also a boon for the modelling experience in general.

You said activities are fundamental. What are they?
Activities are the building blocks for workflows. You create activities much like you may create controls for Forms when you're building a Windows Forms application. There are built-in controls and custom controls and you simply drag them on to your Forms. You can do the same with activities. We have a bunch of activities that come out of the box and they appear on the toolbox when you create a workflow, be it a sequential workflow or a state-based workflow or another type of workflow - you just drag those on to the design surface and compose your workflow. Of course you can also create custom activities and we expect ISVs and other companies to build activities in different application domains. Within Microsoft, the Office group is building a set of activities to support new collaboration and ad-hoc workflow features in the upcoming Office 12. Say an ISV in insurance, where there's a lot of workflow involved, wants to build a claims processing system? Well this process is very unique to each company, so you could imagine the ISV providing a set of customizable domain specific activities for many of the processes. Their insurance company customer would then use these to construct a specific instance of its own claims process.

How would you see this technology affecting human workflows?
We think about "workflow" as including both humans and software systems. Sequential workflow is ideally suited for system based workflows, they're very structured, they have a start and they have an end, they look like a flow chart and effectively you walk down a well trodden path.

workflows, they're very structured, they have a start and they have an end, they look like a flow chart and effectively you walk down a well trodden path.

On the other hand a state-based workflow is well suited for human interactions, where latency and exception conditions are the norm. Making a state machine available at the platform level for every developer to use is a really crucial thing we believe.

What we're doing here is consistent with ASP.NET or WinForms, it's just infrastructure belonging where it should - in the platform. There are many scenarios that you can build on top of this infrastructure and that's the value that you guys add and it'll run just fine on Windows XP, 2003 Server and Vista on the client. One thing to note is that this technology doesn't have its own standalone process nor its own executable; it actually runs in your process so the cost of execution is really low, and it can scale from a Windows console application all the way out to a line of business application or BizTalk Server or SharePoint and those kinds of things by allowing the workflow engine to be hosted as you see fit.

Are any standards being applied that you are aware of for workflow management?
Well, that's always a good question because many companies need to adhere to different standards when building workflow type applications. We've tried to build Windows Workflow Foundation so it can be made to work with different standards without restriction. For example, we've built a set of activities, which you can swap-out from the default activities, which allow our workflow engine to persist to BPEL format and read it back in. The BPEL pack will be available on the Windows Workflow Community website. By making workflow generally and freely available in Windows, we had to avoid a fixed language grammar. I am sure things will be built using WWF that will completely surprise us. Therefore the engine itself literally must not know what each of the grammar steps are in the language sentences made up of activities. When the engine executes an activity that does an "if", a "while", or a "loop", it's the activity's own decision to do whatever it needs to not the engine's. The engine just chains together the set of activities.

the language sentences made up of activities. When the engine executes an activity that does an “if”, a “while”, or a “loop”, it’s the activity’s own decision to do whatever it needs to not the engine’s. The engine just chains together the set of activities. Out of the box we have the activities that you would expect, “if”, “while”, “else”, “compensation transactions” and so on. But you can throw out our activities and still use our engine. You can implement a new standard, or invent your own via custom activities and you can still use the designer to support that. You can host the designer in your own application. And your custom activities would serialize out to XML in a manner that was appropriate with your standard.

If we have the ability to enable Workflow in any application, how does that affect BizTalk Server?

BizTalk Server is an excellent example of a Business Process Management application, and I think this really calls out the difference between infrastructure and product. BizTalk server is an excellent product for creating workflow - effectively “orchestration” - between applications. For example, say I have my Siebel application, my SAP application, and they might be Workflow enabled in themselves but what I want to do is integrate information in those applications together with my trading partners’ information stores. So I have to create a Workflow that sits outside applications and BizTalk Server will absolutely continue to be vital to Microsoft’s strategy in that role. In such scenarios you need to think about messaging, management, adapters, business activity monitoring and all of those standard capabilities of Business Process Management (BPM), all of which BizTalk provides you.

WWF doesn’t provide you with those features - it is a pure form of workflow technology and can be thought of as just the orchestration engine part of BizTalk Server, or BizTalk minus all the BPM stuff.

ctivities, so that future innovations can be enabled to support standards whilst maintaining consistency with the engine itself.

How committed is Microsoft to this stuff?

The key thing is that this technology is going into Windows which means it is going to last for a very, very long time, and we absolutely expect there will be future standards in this space and future standards in each one of the many different scenarios this technology supports. We’ve created an engine that doesn’t run on a fixed language but runs with general activities, so that future innovations can be enabled to support standards whilst maintaining consistency with the engine itself.

This all sounds great! Where can I find out more?

Sure, just point your browser to the Windows Workflow Community website here: www.WindowsWorkflow.net. You can also pick up a great whitepaper on WWF at the Microsoft stand in the exhibition area of JAOO. I am also not surprised that there has been a fair amount of content in the JAOO agenda discussing workflow and orchestration. This area is hot and now WWF has definitely broken the barrier to entry and adoption for application developers.

Have fun with WWF - Thanks!

[Acknowledgements to Scott Woodgate and Paul Andrew, Microsoft Corporation, for the core discussion presented in this interview.]

New presentation

Jan Schoubo , BEA
Business track: 14:00 - 14:30

JAOO IT Run

JAOO IT-Run Team Winners

Aarhus United IT Sprinters
A. Keidser
M. Skovby Andersen
C. Jahn Svinding

ACURE
H. Roulund Andersen
T. R. Møller
J. Jul Jørgensen

IT-Byen Katrinebjerg 4
H. Gregers Jensen
P. Mechlenborg
K. Ligaard Nielsen

Despite the weather 1069 runners joined the run on Tuesday night.