Panel

The Role of Objects in a Services-obsessed World

John Tibbetts (chair)
KINEXIS

Ward Cunningham
AboutUs.org

Carl Lentz
DTE ENERGY

Jeroen van Tyn
DBI CONSULTING

Abstract
Every half-decade or so, the computing world is infected by a meme that energizes IT and stimulates architectural thinking but also distorts discussion and clouds judgment. A few generations ago it was objects; now it's services. As every developer has noticed, SOA is everywhere—even, perhaps, in some places it shouldn't be. What has this focus on services done to the role of objects? Some of the more extreme SOA proponents maintain that service-reuse replaces object-reuse across the board. More mainstream architects view these two models as complementary reuse strategies at different levels of scale. There are even object diehards who think that services can't come close to the flexibility and durability of objects. This panel will represent the full range of opinions. Specifically, panelists and attendees will be challenged to explore such topics as: Where is the scalability boundary between object responsibilities and service responsibilities? Do we have to conceptualize, design, or implement differently at different levels of scale? Where and how do binary components such as RMI, EJB, or CORBA fit in alongside more loosely-coupled interfaces such as web services? Panelists will also be invited to comment on whether some of the new concepts defined into SOA—orchestration and discovery, for example—can be profitably fed back into the object world.

Categories & Subject Descriptors:
D.2 Software Engineering
H.4 Information Technology and Systems
J.9 Computer Applications
K.0 Computing Milieux

General Terms: Design

Keywords: SOA, design, software, architecture, Object-orientation, Service-orientation, Software components

1. John Tibbetts (chair), john.tibbetts@kinexis.com

JOHN TIBBETTS is president of the consulting firm Kinexis, the inventor of the open-source WorkThru framework, and the moving force behind the WorkThru community. He has specialized throughout his 35-year career in the application of object concepts to large transactional systems and has recently become an important theorist/implementer in the area of agent-based workflow and automated collaboration. He is a former columnist for *InformationWeek* magazine, a Senior Consultant at the Cutter Consortium, a technology advisor to numerous startups, a consultant to utility, financial, and telecommunications companies, and an active developer. He is a past OOPSLA keynoter.

POSITION: The best way to think lucidly about services is to think not about 'architecture' but about 'orientation.' Let's remember that the term Service-Oriented Architecture (SOA) was a creation of two Gartner analysts, more skilled in product positioning (a.k.a. marketecture) than in software architecture. I find it much more helpful to lower-case "service-oriented" and use the term to describe not what you buy but how you think.

With this as a starting point, I see service-orientation and object-orientation as two orthogonal reuse strategies that operate at vastly different levels of scale. Objects, small and single-minded, are at the scale of software atoms. Like atoms, they tend to be highly collaborative with other atoms and other types of atoms. Services exist at a much coarser-grained scale, like biological organisms that have an independent existence from one another. Services can collaborate as well as objects can, but in a different way—one that is relatively indifferent as to the inner workings and structure of the other services involved. One challenge for the object practitioner moving into services is to know when and when not to collaborate.

As an implementer, I would not care to you use any approach other than objects for building a new service. But as a service consumer I care little whether the service is written with objects, with scripts, with rules, or, given the right glue, with spreadsheets. To me, this is the fundamental difference between objects and services: Objects are primarily a development-time phenomenon whereas services are primarily a run-time phenomenon. A development organization is object-oriented when it uses OO concepts, practices and languages (which are only visible when one looks under the hood). An organization is service-oriented when it has developed a culture of leveraging existing services that are hosted by theirs or other organizations. In a services-oriented environment, governance—who is responsible for maintaining and extending services being used by other consumers, known and unknown—becomes more important to success than does development practice. And that is a point that many development organizations still haven't absorbed.

OOPSLA'07 October 21–25, 2007, Montréal, Québec, Canada.
ACM 978-1-59593-865-7/07/0010.

2. Ward Cunningham, ward@c2.com

WARD CUNNINGHAM is the Chief Technology Officer of AboutUs.org, a growth company hosting the communities formed by organizations and their constituents. Ward co-founded the consultancy, Cunningham & Cunningham, Inc., has served as a Director of the Eclipse Foundation, an Architect in Microsoft's Patterns & Practices Group, the Director of R&D at Wyatt Software and as Principle Engineer in the Tektronix Computer Research Laboratory. Ward is well known for his contributions to the developing practice of object-oriented programming, the variation called Extreme Programming, and the communities supported by his WikiWikiWeb. Ward hosts the Agile-Manifesto.org. He is a founder of the Hillside Group and there created the Pattern Languages of Programs conferences which continue to be held all over the word.

3. Carl Lentz lentzw@dteenergy.com

Carl Lentz is a technical architect and software developer. He is currently responsible for promoting and deploying a service oriented development strategy and architecture at DTE Energy, a diversified energy company involved in the development and management of energy-related businesses and services nationwide.

Carl has been working in the IT industry for the past 17 years. He started his career in IT at Ford Motor Company, working in a divisional data center as a Computer Operator. For the past 6 years he has worked as a software developer and architect at the company's nuclear power plant working on business support systems. He recently transferred to the Quality Management Group to work with other Technical and Enterprise Architects or a wider range of applications.

Carl is also the process owner for the Technical Solutions aspect of the Solution Delivery Process, an in house developed agile project management methodology. In that role he helps guide and direct the development efforts and processes for both software and infrastructure projects.

POSITION: I work for Detroit's DTE Energy, a company that has a long history of commitment to software engineering best practices. In particular we're unusual in our dual interest in both Agile methods and formal process measurement using the Software Engineering Institute's Common Maturity Model Improvement (CMMI). After a decade of refineing our object-orientation practice we have over the last few years embraced service-orientation as well. This has given us a good viewpoint for seeing where they're similar and where they're different. We look at these comparisons from the point of view of underlying concept, skills, process, testing.

The same development teams build applications, binary components and services. But since we recognize the inherent differences between these tasks, we find that we need to inform the teams on their characteristics with regards to coupling, transaction support, ease of reuse, and reuse benefit. We have refined our formal Software Development Process (SDP) to add new processes to capture service metadata as a development artifact, and we have added a review to consider reuse as either a component or a service. Overall, the development aspects of objects versus services are relatively simple to deal with.

But when it comes to testing, service testing is considerably different than object testing. On the one hand, service tests can be created by non-developers, since the service request and response can be easily captured with a few tools that have been built or bought. This can greatly simplify test construction. On the other hand, we've found a new testing responsibility. Since our services can be used by many different constituencies we find that they require us to support the testing of their own test cases. The service consumer wants to make sure that a subsequent change by the service owner doesn't break some specific function of the service that they need to work.

This brings us to governance issues. The governance of objects and components is relatively straightforward: We create the gadget and put into a repository and fix it when we need to. But a service is only a service if it's running somewhere. Someone in the organization has to be motivated to keep it running, keep it correct, extend its behavior if need be, and keep test instances always available so that new consumers can try them out.

4. Jeroen van Tyn, jeroenvantyn@dbiconsulting.com

Jeroen van Tyn is an Enterprise Architect with DBI Consulting, a business technology consulting firm that specializes in integrating business operations. He is also a Senior Consultant with the Cutter Consortium, an international IT advisory firm. His principal focus is on business architecture and its relationship to enterprise IT architecture and software development methodology. This stems from his repeated observation that, while technological capabilities continue to explode, organizations struggle with applying technology in a way that really matters to their business.

Jeroen has over fifteen years of experience in information technology and business analysis. He has served as enterprise IT architect, business architect, business analyst, project manager, team lead, developer, consultant and mentor on a wide variety of enterprise architecture, software development and business analysis projects for global corporations. He has led and mentored teams in industries ranging from finance, insurance and health care to manufacturing and telecommunications.

Jeroen has delivered formal training in enterprise architecture, requirements management, the Rational Unified Process and object-oriented analysis and design to software development professionals across the U.S. and in Canada. He has been a presenter at a number of regional and

national conferences. Jeroen van Tyn may be reached at the above address or at jvantyn@cutter.com.

POSITION: Let me state at the outset that I speak from within the context of enterprise architecture. That is, I mostly work with large organizations that are trying to get the pieces of their enterprise (business units, processes, systems, information and so on) to function as a coherent whole.

From a technology perspective, services and particularly Service-Oriented Architecture (SOA) are perhaps what objects always wanted to grow up to be. The hype says that we can create large-grained, business-meaningful services that can be dynamically coordinated to quickly construct and deploy business processes that are responsive to ever-changing business needs and innovation. (Buzz-word count, please!)

Objects, despite an early period of bluster and fancy, never quite made the cut in providing easily shareable, enterprise-level functionality – take IBM's San Francisco/Shareable Frameworks project as a notorious example. Services, on the other hand, are specifically supposed to deliver on this.

Here's a dirty little secret: about two-thirds of companies use web services for local application-specific purposes. I'm not making this up: this is based on real research. In other words, a lot of people are using services to do object stuff. Now what's the point of using this standards-based, cross-platform capability in order to build things that by definition will be playing inside the same sandbox?

Now on to the enterprise. To me, SOA is, or should be, inherently an enterprise concern. If the hype-phrase I spun just a minute ago doesn't speak to an enterprise context, then what does? In order for large-grained business-meaningful services to be worth a whit to the enterprise, they must be based upon concisely articulated shared business semantics. In other words, the service-enabled enterprise at a minimum requires a shared enterprise domain model. Now if there's a better way to get at a shared domain model than using an object-oriented approach, I'd sure like to know what that is! SOA thus requires us to apply all of the object-oriented analytical methods we've worked so hard on over the years.

That brings me to my last point: the biggest problem I've seen over and over with object orientation, and now service orientation, has nothing to do with technology. We've got all this great technology: so what? The difficulty always has been and still is, how do you make a "good" object/service? Before you start talking about reuse, you have to be able to create something *useful*. This requires analytical skill and development discipline. Remember "spaghetti-O's", the OO version of spaghetti code? And now the pressure is even greater, because the whole thrust of service-orientation is is centered around the hubris of the dynamically served business enterprise. The analysis required to do that well is *hard*, and no amount of XML, WSDL, virtual machines and application servers is going to do you a bloody bit of good in getting there.