Oxford University Software Engineering Centre

Module on Web Services and SOA

Andrew Martin

August 2006

The course as presently delivered as specified as follows.

Learning Outcomes

Those completing the course will :-

1. understand the notion of a service-based architecture, and be able to describe the strengths and weaknesses of this approach;
2. be able to implement and deploy simple web services using one or more development platforms;
3. be able to define and design applications as combinations of services, with a suitable treatment of security, and be able to discuss the emergent properties of those composite services; and
4. know the research context and some potential future directions for these services and architectures.

Course Contents/Syllabus

The course presupposes a basic understanding of XML schema and XML namespaces. Practical exercises entail a limited amount of Java programming, usually by extending and adapting supplied skeleton code. The design discussion assumes a basic reading knowledge of UML.

Topics:

1. Web-based architectures (1 day)

   Considers the idea of doing distributed computing using the technologies of the World-Wide Web, exemplified by the REST architectural style. Reviews the properties of those technologies (with reference to the underlying protocol stack), and the extent to which HTTP 1.2 successfully implements Fielding's pure REST ideas in any case. The chief shortcoming of the approach may be summed up as saying that the middleware layer is slender, and in places poorly abstracted, so that although REST based services are popular, they are necessarily unlikely to scale very well.

   This is followed up with a practical exercise in which students use and implement REST-style services, and observe the data passing on the wire, using a TCP monitor.

2. Service-oriented architectures (1 day)

   Begins by considering the notion of software construction using a collection of components. Reiterates this using Erl's 'principles of service orientation': Service reusability

– Service contract – Service loose coupling – Service abstraction – Service composability – Service autonomy – Service statelessness – Service discoverability.

Describes the 'publish–find–bind' model, as supported by SOAP, WSDL, UDDI. Emphasis on the first two.

Explore toolkits for writing web services; specifically Axis. Practical exercise using Axis.

3. Structures (half day)

   A discussion, mostly at quite a high level of abstraction, of Message Exchange Patterns; Co-ordination; Atomic Transactions; Business Processes; Orchestration; Choreography; Addressing; Reliable Messaging; Correlation; Notification; Policy. Mention of relevant WS-* specifications in passing, exploring how (tags, protocols, etc.) these achieve their goals.

   Includes a very high level (4/5 slides) consideration of workflow concepts.

   Discussion of state and statelessness. (REST lecture has already made a virtue of stateless services.) This flows into a discussion of Grid services, and a mention of WS-ResourceFramework.

4. Security (half day)

   This topic begins with a brief review of the nature of security requirements, and basic technologies such as encryption and PKI for achieving these.

   We then discuss various ad hoc approaches (such as the web service access tokens used by Google and Amazon), together with HTTP authentication and transport layer security.

   XML encryption and XML signature enable us to discuss message level security, and hence WS-Security etc. We sketch the four-year-old web services security roadmap, and discuss the parts which have actual prospects of becoming concrete. Introduce the idea of security token service, push and pull models for security tokens, generalized into the ideas of PDP and PEP.

   Brief mention of WS-Policy as a means of communicating security policies; SAML as an expressive notation for writing security tokens; XACML as a language for writing detailed access rules. Reflection on whether this level of detail is really feasible or desirable in most application contexts.

5. Engineering Service Oriented Architectures (half day, plus extended practical exercise, incorporating some of the earlier topics also)

   Questions of what a SOA lifecycle might look like. How to describe services; how to test them. UML diagrams for recording workflow, deployment, interactions. (UML as a surrogate for abstract BPEL.) Strength of the 'assembly of services' model.

   Semantic web as an attempt to allow persistent distributed references to concepts and ideas; language for describing services and data (RDF/RDFS/OWL). Semantic web services (OWL-S) as a means to use these ideas for building services.

   Observations on the shortcomings of these ideas for real engineering; comparison with the prospects for using something akin to Z for the task.

6. Interoperability and Prospects (quarter day)

   Largely a wrap-up session. Brief consideration of WS-I as a unifying force for defining de facto interoperability. Discussion of strengths and weaknesses of the WS ideas as they stand; extent to which the early vision can be/is being implemented; extent to which it was amiss. How to avoid building OO systems using SOA; loose coupling, coarse-grained interfaces; few messages. Good and well-defined behaviour in the presence of faults.

Textbooks/Reading List

1. Thomas Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice-Hall, 2005.

   > *Quite a thorough treatment of all the topics; excruciating detail in places, perhaps repetitive.*

2. Gustavo Alonso, Fabio Casati, Harumi Kuno and Vijay Machiraju, *Web Services: Concepts, Architectures and Applications*, Springer 2004.

   > *This is the current course textbook — one of the few genuinely academic texts I've found. It is decidedly verbose, somewhat outdated already, but nevertheless quite a treatment of topics for their own sake, and also showing how the present technologies have been arrived at.*

3. Munindar P. Singh and Michael N. Huhns, *Service-Oriented Computing Semantics, Processes, Agents*, Wiley 2005.

   > *I found this book quite recently: it's gratifying that it takes a broadly similar approach to my own course design! It's quite a large book, but would, I suspect, make quite a good text for the course as it stands now.*

4. Dan Woods and Thomas Mattern *Enterprise SOA: Designing IT for Business Innovation*, O'Reilly, 2006.

   > *Definitely at the 'business' end of the topic; heavily influenced by the BEA approach to SOA.*

5. Eve Andersson, Philip Greenspun, and Andrew Grumet *Software Engineering for Internet Applications*

   > *An MIT undergraduate text; too basic (or too web-oriented; not enough SOA) for our task here. Also one of the most edgy textbooks I have ever encountered.*