The All-Singing, All-Dancing Business Process

Doug Tidwell
IBM Corporation

dtidwell@us.ibm.com
Agenda

- Standards roll call
- Application development with SCA
- Data marshalling with SDO
- User interfaces with XForms
- Workflow with BPEL
The big picture

- We’ll describe a business process that handles purchase orders. The process uses:
  - Workflow
  - Service-oriented architecture
  - Data-driven user interfaces
Standards roll call
Standards roll call

- Our discussion and demos are built around four open standards:
  - SCA
  - SDO
  - XForms
  - BPEL
Standards roll call

- **SCA** is the mechanism for invoking services. We can change the services used by the process without changing the application code.
- **SDO** gives us a flexible data model for integrating different kinds of data from different sources. We can make changes to the data model without breaking our application.
SCA and SDO were developed by osoa.org:
The specifications work of OSOA has been turned over to OASIS.

Open CSA is the Open Composite Services Architecture group.

See oasis-opencsao.org for more details.
Standards roll call

- **XForms** ties our UI to the data structures. We can use XSLT and CSS to create the UI from the data structures; if the data structures change, the interface is simply regenerated.

- **BPEL** describes the business process. Steps in the process can invoke services or human tasks. We can change the process by changing the BPEL description.
Standards roll call

- XForms is a project of the W3C.
  - Version 1.1 is at CR
  - Version 1.0 (3rd edition) was released last October.
- BPEL (WS-BPEL) is a project of OASIS.
  - Version 2.0 was released last April.
Other standards

- XML Schema – Defines the data structures.
  - The data passed back and forth by SCA is defined by XML Schema (maybe in WSDL)
  - The data objects used by SDO are defined by XML Schema
  - The user interface built by XForms references elements defined in the XML Schema
  - The data passed back and forth by BPEL is defined by XML Schema (in WSDL)
Other standards

- **WSDL**
  - An SCA component or composite can be a Web service based on a WSDL file.
  - The partner links used in BPEL are defined in WSDL files.
Implementations

- We’re using several open-source tools:
  - Apache Tuscany (SCA and SDO)
  - Mozilla XForms Plugin
  - Google Ubiquity XForms library
  - Apache ODE (BPEL)
SCA

A better way to build applications in an SOA
A quote

- Anyone who’s interested in the future of application development should also be interested in SCA.

  ➢ David Chappell, *Introducing SCA*
  davidchappell.com/articles/Introducing_SCA.pdf
SCA

- A more elegant programming model for composite applications.

[My other presentation is 90 minutes of SCA goodness.]
Using a component

- When dealing with a component (in an SOA or not), there are three important pieces of information:
  - The **interface** of the component
  - The **implementation** of the component
  - The **access method** to invoke the component

- We’ll consider how we use this information to invoke components.
The bad old days

- Originally, most components were hardwired into an application:
  - The application knew the details of the component’s interface at build time.
  - The application accessed the component’s implementation at build time.
  - The application knew the details of the component’s access method at build time.
- This worked (and still does), but the application is brittle.
Web services

- SOAP introduced a way to invoke a remote service with an XML envelope.
  - The application knew the details of the component’s interface at build time.
  - *The application did not access the component’s implementation at build time; the component is invoked at run time by the SOAP infrastructure.*
  - The application knew the details of the component’s access method at build time (usually SOAP/HTTP).
An SCA application is even more dynamic:

- The application knows the details of the component’s interface at build time.
- The application does not access the component’s implementation at build time; the component is invoked by the SCA invocation framework.
- The application does not know the details of the component’s access method at build time; this is also handled by the SCA invocation framework.
SCA

- SCA gives you a **single programming model for using services**.
- In SCA, everything looks like a POJO.
  - `myService.foo(x, y);`
- Without SCA, you have to learn more and more interfaces.
  - In Java alone, you might have EJBs, RMI, JCA, JAX-WS or JAX-RPC.
One way to look at SCA is that it takes all of the details of access methods, implementations, encryption, authentication, etc. and moves them into the middleware layer.

- Application developers write business logic, code that actually builds value for your organization.
- The details of using services are handled by SCA.
- As the details change, your applications (and the developers who wrote them) aren’t affected.
SCA simplifies governance

- With SCA, you define a component one time, in one place, then point your applications to that definition.
  - If all of the applications use the same definition, you know what components your organization uses.
  - If you need to change the component or how it works, you make that change one time, in one place.
Demo

- We’ll look at a brief demo of an SCA component that calculates shipping charges.
A worldwide phenomenon

Malmö, Sweden, 3 July 2008
SDO
Service Data Objects

- **SDO gives you a single API to a wide variety of data sources.**
  - Similar yet incompatible data binding frameworks will outnumber the human population by 3Q 2013.
- You and I as developers focus on create / replace / update / delete operations.
- We don’t know what the data source actually is. (JDBC, XML database or XML file, JMS, Web service, JCA)
The goal of SDO

- I have some data.
- I use the data wherever and however it’s stored (RDBMS, XML file, LDAP, etc.)
- I use the most convenient language for CRUD operations on the data (SQL, XQuery, modified XPath, etc.)
A disconnected interface to data sources

- Application
- Data Access Service
- Data graph
- Data object
- Change summary

Connections:
- JDBC to RDB
- XPath/XQuery to XML DB
- Local to EJB
- XML/HTTP to Web service
- CCI/Proprietary to JCA
SDO concepts

- The data itself is stored in a data object.
- A hierarchical set of data objects is represented with a data graph.
- The data object’s schema definition is stored in the data object metadata.
Data objects

- A **data object** is just a set of properties.
  - Properties can be primitive types, multi-valued fields and other data objects.
  - A data object provides get and set methods for its properties. You can also use XPath to access the properties.
Data object APIs

- In SDO, data objects have both dynamic and static APIs.
  - A dynamic API looks like this:
    ```
    object.
    setString("firstName","Doug");
    ```
  - A static API looks like this:
    ```
    object.setFirstName("Doug");
    ```
Data graphs

- Because SDO is a disconnected data model, we need to keep up with any changes to the data.

- A **data graph** contains the details of:
  - Any change to the properties of an object, including a reference to the changed object, the property that was changed, and the old and new values
  - Any data objects added to the graph
  - Any data objects deleted from the graph
Data object metadata

Along with data objects and data graphs, SDO uses **data object metadata** to store the object’s schema.

- This includes datatypes of its properties, relationships between fields, constraints, etc.
SDO and XML

- SDO integrates well with XML.
- Every data exchange in the Web services world is done with XML (SOAP messages), so this makes SDO a great technology for working with the many data sources an SOA might use.
XML and relational

- The Tuscany SDO implementation has several nice features:
  - Create a data graph from XML
  - Create a data graph from an RDBMS
  - Create SDO metadata from an XML Schema
  - For a given SDO, serialize its data as XML and serialize its metadata as XSD
Schema flexibility

- SDO lets you change data objects without breaking your code.
  - Our purchase order process contains information about a customer.
  - If the customer is a Java object, any change to the Java object is fatal.
  - With SDO, we can change the object without breaking the code.
Demo

- We’ll look at a brief demo that uses SDO to create a PurchaseOrder data object from the XML Schema.
- Another demo reads the schema into a PurchaseOrder data object.
- Finally, we’ll change the schema without breaking the reader.
XForms

Data-driven user interfaces
XForms

- Forms are integrated with data models
- One form can be rendered on many types of devices
- Forms use a declarative model
- **XForms has an underlying data model (XML) that can be generated from your existing business objects.**
With XForms, we’re specifying what the data is.

- The user can choose one item in the list
- For each item, there is a label and a value
- The label for the selection and the labels for each item can be extracted from an XML document.

We *don’t* say how this is displayed.

- We can define rules elsewhere to say how this is rendered based on the number of choices, type of device, whatever.
Deploying XForms

- One of the challenges of deploying XForms is getting XForms support on the client.
- There are two approaches:
  - Requiring a browser plug-in
  - Using Ajax controls
Using a browser plug-in

- Typically useful in an intranet
  - For Internet Explorer, there’s MozzIE, written by Peter Nunn. See [http://sourceforge.net/projects/mozzie](http://sourceforge.net/projects/mozzie).
Using Ajax controls

- The Chiba project provides an open-source XForms engine.
  - It uses XSLT to transform the XForms document into Ajax-enabled HTML pages and send them to the client.
  - Chiba doesn’t require any client-side plug-ins.
Using Ajax controls

- The Orbeon Presentation Server is an open-source XForms engine based on Chiba.
  - Like Chiba, Orbeon doesn’t require any client-side plug-ins.
  - See http://orbeon.com/.
Using Ajax controls

- The XForms Ubiquity project is designed to run XForms on the client.
  - A JavaScript library handles everything on the client device.
BPEL

Workflow ahoy
The basics of BPEL

- **BPEL** (*full name*: Web Services Business Process Execution Language) is an open standard for defining workflows.
  - The standard lives at OASIS: [http://www.oasis-open.org/committees/wsbpel](http://www.oasis-open.org/committees/wsbpel)

- Version 1.0 was released by BEA, IBM and Microsoft in July 2002.
  - It was called BPEL4WS at the time.
The basics of BPEL

- BPEL Version 2.0 was approved by OASIS on 12 April 2007.
  - Dozens of companies participated, including BEA, IBM, Iona, Microsoft, Oracle, Red Hat, Rogue Wave, SAP, Sun and Tibco.

- BPEL4People adds more sophisticated support for human tasks.
  - IBM, SAP, Oracle, Adobe, Active Endpoints
BPEL elements

- Starts the BPEL process – we receive a request
- Ends the BPEL process – we send a reply
- Invokes a Web service
- Moves data from one variable to another
- A small piece of code embedded in the BPEL process
Other BPEL elements

- BPEL supports tasks in both sequence and parallel (or a combination).
- BPEL has a choice construct (switch).
- Exception handling: Throw, rethrow, compensate, terminate.
- Pick: Wait for one of a number of events to happen.
- No-op: Useful for synchronization...
BPEL elements

- **Interface partners:**
  - Definition of an interface (typically WSDL without a binding)

- **Reference partners:**
  - Implementations of particular interfaces (Web service endpoints)

- **Variables:**
  - Manage state information
  - Typically map to request and response messages for Web services
The process flow

- OrderHandlingFuture1_Flow
  - OrderHandling (Future1)Receive
    - Check Order Handling Policy for Automatic Approval
      - Check Customer Account Status
      - Review Order
        - Ship Order to Customer
        - Cancel Order and Send Notification
      - Update Order Database
Implementing the process

- Now that we’ve defined our process, we need to link the various steps in the process to the appropriate Web services.

- For each link, we need:
  - An interface partner (defines the WSDL interface)
  - A reference partner (the implementer of the Web service)
  - One or more variables (typically a request and a response)
Adding a human task

- Real-world processes typically require human interaction at some point.
- BPEL can have human task as part of the main process flow.
  - Auditing, security, judgement calls, etc.
- The recently released BPEL4People spec adds more capabilities for adding human tasks to a BPEL 2.0 process.
BPEL and SCA together
The big picture
The big picture

- We have a business process that handles purchase orders.
  - BPEL defines the steps in the process, including the services we use and the human tasks required.
  - SCA invokes the services.
  - The data in the purchase orders, product descriptions and customer accounts is SDO.
  - The user interfaces to the process, the human tasks and the order entry system itself are built with XForms.
We have a business process that handles purchase orders.
- BPEL defines the steps in the process, including the services we use and the human tasks required.
- SCA invokes the services.
- The data in the purchase orders, product descriptions and customer accounts is SDO.
- The user interfaces to the process, the human tasks and the order entry system itself are built with XForms.
Resources
Resources

- Apache Tuscany

- Mozilla XForms Add-on

- XForms Ubquity

- Apache ODE
  - [ode.apache.org](ode.apache.org)
Eclipse BPEL project

- Visit [eclipse.org/bpel](http://eclipse.org/bpel)
- Easiest install path:
  - Download and unzip Europa (Eclipse 3.3)
  - Help ➔ Software Updates ➔ Find and Install
- Great developerWorks article on the BPEL tooling:
Summary
Summary

- Our business process defines the workflow involved with processing purchase orders.
- The process is designed to be as flexible as possible:
  - We can change the services invoked
  - We can change the data structures
  - We can regenerate the UI
  - We can change the workflow
My other session

- Service Component Architecture: A Better Programming Model for SOAs
- 90 minutes of SCA goodness!
  ➢ Friday @ 8:30, Torreys Peak 3 & 4
The holidays are coming!

- The second edition of *O’Reilly’s XSLT* is available! (ISBN 0-59652-721-7)
- A great gift for:
  - Freemasons
  - Expectant mothers
  - Supermodels
  - Disaffected loners
  - Your parole officer
Thanks!

Doug Tidwell
IBM Corporation

dtidwell@us.ibm.com